National Library of Canada
Bibliothèque nationale du Canada

Canadian Theses Service
Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, tests publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30.

NL-339 (r. 88/04)

Canada

THE UNIVERSITY OF ALBERTA

Microprocessor-Based 3D Reconstruction from CT Images for Neurosurgical Applications

by

Kurt Klingbeil

©

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE

OF Master of Science

Department of Electrical Engineering

EDMONTON, ALBERTA

FALL 1988

Permission has been granted to the National Library of Canada to microfilm this thesis and to lend or sell copies of the film.

The author (copyright owner) has reserved other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without his/her written permission.

L'autorisation a été accordée à la Bibliothèque nationale du Canada de microfilmer cette thèse et de prêter ou de vendre des exemplaires du film.

L'auteur (titulaire du droit d'auteur) se réserve les autres droits de publication; ni la thèse ni de longs extraits de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation écrite.

ISBN 0-315-45675-2

THE UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR Kurt Klingbeil

TITLE OF THESIS Microprocessor-Based 3D Reconstruction from CT Images for Neurosurgical Applications

DEGREE FOR WHICH THESIS WAS PRESENTED Master of Science

YEAR THIS DEGREE GRANTED FALL 1988

Permission is hereby granted to THE UNIVERSITY OF ALBERTA LIBRARY to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without the author's written permission.

(SIGNED) Kurt Klingbeil

PERMANENT ADDRESS:

9736-80 Avenue
Edmonton, Alberta
T6E 1S7 Canada

DATED August 8 1988

THE UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research, for acceptance, a thesis entitled Microprocessor-Based 3D Reconstruction from CT Images for Neurosurgical Applications submitted by Kurt Klingbeil in partial fulfilment of the requirements for the degree of Master of Science.

.....

Supervisor

.....

.....

Date August 8, 1988

Abstract

A semi-portable microprocessor-based computer graphics system enhances computed tomography (CT) images and displays a three-dimensional model of the essential physical structures of a human brain. This system is intended to aid a neurosurgeon in interpreting CT data to both diagnose the nature and extent of a pathology and to plan possible surgical procedures.

The 3D model is derived from a series of CT slices via image enhancement operations, automatic and manual extraction of contours, and generation of polyhedral surfaces composed of triangular tiles. Movie.BYU is a commercially available software package which provides a full range of graphical attributes with which to display the model.

Detailed operating instructions and notes on the Logical Microcomputer Company (LMC) MegaMicro computer are provided.

Table of Contents

Chapter — Page

I. Introduction 1

II. Three Dimensional Medical Image Data 4

A. 3D Measurements 4

B. Analysis and Display 5

C. Surgical Applications 8

III. The Reconstruction Problem 11

A. Model Selection 11

B. Edge Detection 12

IV. Model Generation 16

A. Data Input 16

B. Image Processing 18

Filtering 18

Histogram Operations 20

False Colouring 21

C. Automatic Contouring 22

D. Manual Intervention 24

E. Inter-slice Registration 25

F. Triangulation and Display 26

V. System Hardware and Software 27

A. System Hardware 27

Computer 27

Graphics Output 27

Graphics Input Device 28

B. System Software and Support 28

Operating System 28

LMC Installation 30

VI. Application Software 32
A. Graphics Boards Utilities 46
B. Cursor 55
C. Colours 56
Command Summary 57
D. Enhance 58
Shell Command Summary 58
Macros 59
Filtering Functions 61
Histogram Functions 63
E. Premos 64
Command Summary 65
F. Registrate 68
G. Movie.BYU 68
Mosaic 70
Display 70
H. Device Drivers for Movie.BYU 73
VII. Sample System Session 78
VIII. Conclusions 96
References 98
Appendix 1 102
Appendix 2 104
Appendix 3 106

List of Figures

Figure		Page
1.	Format of software structure diagrams.	34
2.	Structure of reconstruction system software.	35
3.	Structure of 'enhance'.	36
4.	Structure of 'filter_select' (enhance).	37
5.	Structure of 'histogram_select' (enhance).	38
6.	Structure of 'macros:' (enhance).	39
7.	Structure of 'premos'.	40
8.	Structure of 'aim:' & 'update:' (premos).	41
9.	Structure of 'printstuff' (premos).	42
10.	Structure of 'registrate'.	43
11.	Structure of 'mosaic' & 'display' (Movie.BYU).	44
12.	Structure of graphics commands 'gfx_cmds'.	45

List of Plates

Plate		Page
1.	Peano Curves in 2 and 3 Dimensions	76
2.	Peano Curve Traversal of a Colour Cube	76
3.	Complete Set of CT Images	80
4.	Complete Set of CT Images - False Coloured	80
5.	Four Selected CT Images - False Coloured	81
6.	Single Greyscale CT Image - Histogrammed	81
7.	Single Contrast Enhanced CT Image	83
8.	Edge-enhanced CT Image	83
9.	Difference of Gaussian Filtered CT Image	84
10.	Sobel/Absorption Edge Detected CT Image	84
11.	Auto-extracted Contours overlaid on CT image	85
12.	Auto-extracted Contours on blank background	85
13.	Manually Entered Contours	88
14.	Premos Contours - Stereotactic Locator Pins	88
15.	Premos Contours - Complete Set	89
16.	Premos Contours - Auto Registered	89
17.	Movie.BYU Display of 3D Model	90
18.	Movie.BYU Display with Enhanced Colour	90
19.	Complete set of CT images - Patient #2	92
20.	Movie.BYU Display of 3D Model - 4 Views	92
21.	Movie.BYU Display of 3D Model - with menu	94
22.	Movie.BYU Stereo Pair Display of 3D Model	94

I. Introduction

Computed Tomography (CT) is an X-ray imaging process which produces axial cross sections of the object being examined. A single CT image or "slice" is a two-dimensional representation of the density distribution over a thin cross section through the object.

In medical applications, the internal structure of the body, the organs and any pathology present, are the objects of interest. Normally, internal features can be visually identified on a slice from their characteristic densities and their physical shapes and orientations. Abnormalities in these may offer clues as to the nature and cause of the medical problem. Image processing techniques can enhance subtle features and help to make the interpretation of the 2D images more objective. These techniques include filtering, contrast enhancement, false colouring, and boundary detection among others.

A series of axially displaced 2D slices constitutes a 3D representation of the object. A three-dimensional visualization can be mentally synthesized from a set of slices, but a computer can make this process more objective and quantitatively precise through graphic display of a model derived from the slices. An intuitively natural way to define a 3D model is to describe contours around the features of interest, slice by slice, and then create a surface lying on these sets of axially displaced contours.

A subjective review of the variety of approaches to the three dimensional reconstruction problem discussed in the literature [1,2,3,4,5,6,7,8,9,10,11,12] reinforced this natural inclination.

In practical application to neurosurgery, enhancement of the CT slice images, and measurement and display of a 3D model of the intra-cranial structures can aid in diagnosis. The ability to measure physical characteristics of the model and to relate model features to the coordinate system of a stereotactic frame would aid in planning and carrying out surgical procedures such as biopsy, fluid aspiration, etc. Further, the ability to interactively manipulate radiation and/or thermal dosimetry models and overlay these data onto brain model features would aid in planning and carrying out surgical procedures using implanted radioactive beads and/or laser light.

The purpose of this project was to produce a relatively inexpensive semi-portable computer graphics system which would perform as many of these functions as possible and be easily operable by someone without extensive experience with computers. The system's portability would enable its use in the surgeon's office and in the operating room. The main focus and work of this project was to develop and/or implement computer programs which could achieve these objectives.

The early decision to use an existing computer program for the actual display of the 3D model led to the selection

and implementation of Movie.BYU [13], a software package for manipulating and displaying geometric data which is readily compatible with the tiled-surface contour-based model approach that was chosen.

The other aspects are dealt with by a number of programs written by the author including *cursor*, *colours*, *enhance*, *premos*, *registrate*, as well as a library of basic graphics functions.[1] Their source code listings have not been included in this document, but are available in printed or electronic form.

---

[1] Throughout this document, *italicized terms* refer to application programs, operating system commands, or files.

II. Three Dimensional Medical Image Data

Modern medical practices rely increasingly on sophisticated measurements of the properties of body tissues. Diagnosis, and possibly treatment, depend on accurately "seeing" what the situation is. Advances in virtually non-invasive medical imaging systems are particularly important to neurosurgery since manual exploration and manipulation of the brain are considerably more dangerous than such surgery in the rest of the body.

Neurosurgery and neuroradiology were revolutionized by the introduction of CT scanners in the early 1970's [14]. Advances continue to be made in imaging systems and in surgical techniques, but obtaining and understanding precise information about what is inside the patient's body remains as their foundation.

A. 3D Measurements

Sensitive detectors can measure the intensity of radiation coming from a body. Depending upon the imaging technique employed, this energy might be unabsorbed X-rays from a beam directed through the body, emissions from artificially introduced radioactive substances, naturally occurring particle emissions, or re-emitted radio frequency waves.

In most imaging systems, digital computers control an electro-mechanical apparatus which determines the locations, directions, and timing of the detection of this energy. The

computers also collect the data and then perform complex mathematical and heuristic manipulations to reconstruct the internal arrangement of patterns of absorption or emission.

For example, an ordinary X-ray image, captured on film, shows the total absorption by body tissues along each path from the X-ray source to the film. No information about the depth within the body is available from a single image. A collection of such images, however, taken from all directions around the body, contains the basic information needed to calculate the absorption and hence the density of each point within the body. This is the basis for Computed Tomography, a process which yields cross sectional images of tissue density.

Positron Emission Tomography and Ultrasound are two other imaging processes which yield cross sectional images of tissue properties. A collection of axially displaced cross sectional images make up a three dimensional array or image of tissue measurements. Nuclear Magnetic Resonance is an imaging process which uses strong magnetic fields and radio waves to produce a three dimensional array directly.

B. Analysis and Display

Having measured and calculated a 3D array or image, presenting this voluminous data in easily understood forms poses a problem. Flynn et al. [1] examined the basic system requirements and a number of display and analysis methods. The need to extract simple descriptions of the features of

interest and thereby drastically reduce the amount of data actively manipulated is a common thread in Flynn's and other works.

In neurosurgical applications, the outlines of the skull and some of the brain's physical structures are the features of interest. Ventricles are the well-defined fluid-filled spaces between the folds and lobes of brain tissue. Tumours or lesions consist of abnormal or diseased tissue.

These intra cranial structures can be described by polygonal surfaces lying on cross sectional contours [6,8,11,12] or by directed graphs of unit volume elements called voxels[2] [2,3]. Finding the boundaries between regions of interest is of primary importance to any method. When image contrast is high, the boundaries can be found by simple automatic algorithms. Low contrast images require sophisticated algorithms or may yield only to subjective interpretation and manual edge-tracing by experienced operators.

Other analysis methods that might be emloyed are texture analysis of the image and statistical comparisons of assymmetries in the shapes of the ventricles, or of the skull. They may highlight potential current or future problems [11].

Once a model of the patient's features of interest has been extracted, it may be graphically displayed in a number

---

[2] Voxels are the 3D analog of pixels.

of ways, including:

1. Cross-sectional line-drawing views of the model, either on plain backgrounds, or overlaid on the original continuous tone images. No 3D simulation is achieved, but intermediate stages in the building of a 3D model may be monitored and controlled.
2. Perspective wire frame views of the entire model with 3D simulated by perspective and removal of hidden lines. Image quality is sacrificed for speed of computation.
3. Shaded surface views of the model which provide a more realistic appearance at the expense of increased computation. Iterations of wire frame views are usually used to set up the desired viewpoint for a shaded surface view.
4. Stereoscopic pairs of wire frame or shaded surface views. The 3D simulation is strongly enhanced by the effects of parallax.
5. Virtual 3D image of model by projection of computer-synthesized holograms or CRT images synchronized with a computer-driven vibrating mirror. Virtual displays may become more common in the future, but probably only on the largest systems.

All but the last of these have been implemented on this system both in monochrome and in colour.

C. Surgical Applications

Apart from the information provided by CT scanners and other imaging systems, the stereotactic frame is perhaps the most important tool available to neurosurgery. It provides the means to go beyond diagnosis to relatively unobtrusive intervention. Prior to the mid-1970's, surgery was limited to the most severe cases or to those where the tumour lay close to the skull. Now, such procedures as biopsies or fluid aspirations to reduce intra cranial pressure are simple and routine. Even deep-seated tumours have been successfully removed using a specially modified frame, a laser, and a computer graphics system.

Stereotactic frames have been in use since the early 1900's, though then mostly for animal experimentation. Image to frame registration was done by manual triangulation from planar X-ray images. The introduction of CT scanners has spurred the development of frames to the present state in which frame registration is automatically calculated in about 20 seconds, and instrument placement to submillimetre accuracy is attainable.

A modern stereotactic frame consists of an arrangement of precision gimbals, slides, linear guides/probe holders, and locator pins on a base assembly which can be securely fastened to a patient's skull.

Prior to surgery, the patient is scanned with the locator assembly of the frame in place. The locator pins appear on the tomographs and are used to calculate the

various frame angle and displacement settings needed to hit the target area. During surgery the frame guides surgical instruments to their intended position within the skull. A small drill bit, guided by the frame, makes a hole through the skull. Other instruments are then guided through this hole to the specified targets.

Computer graphics treatment planning systems are available and are continually being further refined. They enable the simulation of surgical procedures, such as external radiation therapy using particle beams or the implantation of radioactive pellets which can deliver extremely high doses of radiation to small areas within the brain. Placement of the radiation sources can be optimized to maximize destruction of the pathology while minimizing exposure of the surrounding healthy tissue.

Some systems incorporate the stereotactic anatomical database - i.e. radiographs, stereotactic arteriography, and stereotactic atlases - into their programs[15]. This information, combined with the CT data for a particular patient yields a very accurate picture of that particular brain. Extra precautions can then be taken. For example, to guard against damaging an artery, and thereby causing intra cranial bleeding,[3] an artery-avoiding stereotactic trajectory to the target can be calculated and executed.

Infra red laser irradiation therapy technology is also progressing and will depend on computer graphics systems and

---

[3] Rhodes et al. [14] cite this concern as one reason for performing a post-operative scan.

stereotactic frames for dosage calculation and control.

Deep-seated intra cranial tumours have been successfully removed using a surgical carbon dioxide laser, a specially adapted stereotactic frame, and a computer graphics system[15]. The graphics system presents a view of the tumour outline in relation to the laser's focal point as the tumour is vapourized layer by layer. The system does not utilize 3D reconstructions, but rather, simply keeps track of the positions of the surgical instruments in the CT and stereotactic coordinate systems as the operation progresses through successive layers.

III. The Reconstruction Problem

In this application, the skull, the ventricles, and the tumours or lesions are the main features we are interested in. We want to re-construct a 3 dimensional model of these features based on a set of CT slices.

A. Model Selection

Two types of models predominate in the literature: the cuberille-based and the tessellated (tiled-surface) contour-based models.

A cuberille-based model consists of a set of unit volume elements (called voxels or cuberilles) located at the boundaries of the objects[2,3]. This model has the advantage that once it has been generated, all model surfaces are fully specified right down to the level of the smallest-element (pixel) without any further calculations. In addition, any surface which can be extracted can be rendered. The data is usually arranged in structures called directed graphs so that it can be efficiently accessed. Display consists of a projection into a pixel intensity array which is subsequently sent to the display device.

The hidden-surface problem is solved by a technique known as depth-sorting, or Z-buffering. This consists of maintaining an extra array of the Z-coordinates or distances from each pixel to the observer. The intensity and Z-coordinate of any given pixel is updated only if the new values were to lie closer to the observer than the old ones.

The disadvantages of the cuberille-model include very large data array sizes, somewhat grainy surface rendition, and the lack of available software to perform any of the necessary reconstruction or display functions.

The contour-based tiled-surface models have the advantages of small data array sizes, good surface rendition, and the availability of software to perform part of the reconstruction and all of the display functions. The main disadvantage is the high number of calculations required to do the perspective projections. Since the data array is so sparse, consisting only of the vertices of the surface tiles, the ways the surfaces intersect and obscure each other have to be calculated for each projection.[4] In addition, some particularly convoluted surfaces may not lend themselves to this approach[2].

B. Edge Detection

Edge detection is a fundamental image analysis problem and a good deal of research has been and continues to be focussed on it. Levialdi [16] has classified the wide range of approaches into five categories:

1. local - examine a small area immediately surrounding each edge pixel,
2. regional - examine larger areas for general trends, textures, etc.,

---

[4] The Z-buffering technique is also applicable in this situation, but it has not been incorporated into Movie.byu.

3. global - applied uniformly to the entire image, e.g. thresholding,
4. line-following - points known to lie on boundaries are linked together with other similar points to build contours, and
5. others... - often incorporating combinations of the above as well as a variety of new approaches.

The local approaches are the most commonly used due to their ease of implementation and their good performance. Current local methods often rely on conventional convolution operators which calculate gradient, Laplacian, or multiple-order derivatives [17,18,19,20,21,22,23,24,25,26]. Most edge-detectors use variations and combinations of these as well as thresholding, non-maximum suppression, pre-filtering, and absorption[24] operations. The other approaches often rely on and/or dynamically control local methods.

The simplest type of edge detection involves thresholding the image and then tracking the non-zero pixels. This yields good results only for relatively high contrast edges such as between tissue and bone or air. In lower contrast areas, gaps may appear or the edges may disappear altogether. Edge connectivity is reduced as a result.

A number of local edge detectors were examined and subjectively evaluated, including:

1. Difference-of-Gaussian (DoG) which consists of

subtracting the results of two different low-pass or averaging filters [17], for example, a 5x5 subtracted from a 7x7,

2. Prewitt, a direction-dependant 3x3 convolution array[5] of the form {1,1,1, 0,0,0, -1,-1,-1},
3. Sobel, a direction-dependant 3x3 convolution array of the form {1,K,1, 0,0,0, -1,-K,-1}, where K usually equals 2, but may range from 2 to 3 according to some sources,
4. Kittler's absorption algorithm [24] which is based on a diagonal superposition of Sobel detectors,[6] and,
5. various 3x3 direction-independent highpass kernels.

The magnitude, direction[7], connectivity, and smearing of the edges in the resultant images were examined. The Sobel detector with parameter K = 5/2 to 3, and the absorption detector with parameter K = 3 offered the best performance.

Three versions of the DoG filter, implemented as the differences between pairs of averaging filters of dimensions 3x3 & 5x5, 3x3 & 7x7, and 5x5 & 7x7 induced a lot of edge smearing and failed to adequately highlight the subtle features.

Prefiltering with an edge-enhancing filter consisting of a rotationally invariant 3x3 high pass kernel of the form {-1,-1,-1, -1, 9,-1, -1,-1,-1} added to a scalar transfer

---

[5] A convolution array is often referred to as a 'kernel'.
[6] See Appendix 2, p. 104 for this kernel.
[7] where applicable; i.e. only for directional detectors

function produced a visually significant effect but appeared not to improve the performance of any of the algorithms except the DoG.

The relationships between the local, regional, and global image characteristics are usually too complex to be totally automated. Typically, following some local edge-detection operations, regional, global, and line-following approaches are employed to attempt to classify the edge pixels and to trace contours joining the significant ones.

A computer graphics reconstruction system should provide functions which can be combined to automate these processes as much as possible, and which can serve as powerful tools for the inescapable manual processes.

IV. Model Generation

A. Data Input

The image data consists of CT images of a patient's brain. In our case, these images are on a sheet of radiographic film output by a CT scanner. Each CT image is digitized by the video camera and the graphics boards and then written to a disk file.

Care must be taken to ensure that the film is uniformly back-illuminated, that the axis of the video camera is perpendicular to the plane of the film, and that the focus, magnification, and horizontal alignment is optimum.

Non-linearity in the camera has not been fully compensated for. The illumination intensity and/or the f-stop of the lens can be used to partially compensate for this. Camera and digitization noise, estimated at 1.5 bits, could be reduced by averaging a number of digitized frames together.

The literature describes other commonly used methods of entering data into the computer including:

1. manually digitizing the contours directly from the film using a graphics tablet,
2. electronically transferring the data between computer systems via magnetic tape, disks, or a wire link, or
3. implementing the reconstruction system on a CT scanner.

The first of these is not very attractive since the 2D images are not available to the system for display or

manipulation, and editing the contours may require a return to the physical film images.

The other two offer advantages of increased data accuracy and resolution, decreased noise, freedom from the greyscale windowing employed in producing the film images, and from the characteristics of the video camera. They also eliminate the manual fiddling around with cameras and bits of film etc.

It may be difficult to gain access to a scanner's hardware and software. This would be required to establish a wire link[8] from the scanner to another computer system. All scanners are set up for data output onto magnetic tape, but the addition of a 9-track tape drive to this system would add significant expense and may substantially reduce portability. This aside, data format compatibilities for each type of CT scanner to be used would have to be established.

A reconstruction system can be implemented on the hardware of a CT scanner provided detailed documentation of the hardware and software, and significant amounts of development time on the scanner are available. Some researchers advocate and conduct surgical procedures in which pre-operative scanning, treatment planning, stereotactic surgery, and post-operative scanning are conducted in the CT scanner suite in single sessions lasting from 30 to 60 minutes [14]. This is practical only in

---

[8] such as a tap into a photoplotter connection, for example

situations where use of the scanner is relatively low or where a dedicated scanner is available.

Using a camera/digitizer ensures compatibility with any CT scanner. The subjective interpretation involved in current diagnostic methods depends upon the visible features in the CT images. These features appear to be adequately recovered by this method.

B. Image Processing

The chief purpose of image processing in this context is to make the boundaries between various regions of similar pixels easier to find. To this end, various methods can make similar pixels more similar, and/or dissimilar pixels more dissimilar.

Filtering

A wide range of filtering functions and the effects they produce (or are intended to produce) are discussed in the literature. They include spatial convolutions and frequency domain approaches using Discrete Fourier Transforms (DFT's).

Fourier transform methods are inherently spatially invariant and entail a fixed minimum overhead. They are most effective for functions which operate on texture and other spectral characteristics, or for transfer functions which would require large convolution arrays[9]

---

[9] The DFT/convolution break-even point is at about a 13x13 kernel if implemented by convolution (Verhagen [27]).

Convolutions are far more commonly used because of their computational efficiency and transfer-function flexibility. In a convolution operation on an image, the final value of any given pixel is determined by its initial value and the value of the pixels in a small window surrounding it. The relation may be linear (i.e. averaging or differencing) or non-linear (i.e. thresholding, local maximum/minimum, median, standard deviation, etc.). The relation may also be either spatially invariant, or else depend on the characteristics of regions of the image (i.e. histograms can provide global or local thresholds). The complexity of the relation and the size of the window should be minimized.

The desired effects of filtering are often either smoothing and reduction of noise (low-pass), or identification of edges and other discontinuities (high-pass). These two processes are at odds with each other and must be balanced.

Certain edge-detecting algorithms are very sensitive to noise. While smoothing the image first blurs the edges somewhat, it also prevents the high-pass functions from being overcome by noise and producing only a randomly speckled pattern.

Experimentation is usually necessary to determine the best filter(s) to use to achieve the desired effects under constraints of processing time and minimization of side

effects.[10]

Histogram Operations

A histogram consists of an array of values each of which contains the total number of members of some set of interest which are equal to that value's index in the histogram array. The cumulative distribution function (CDF) is the integral of the histogram scaled down by the total number of members in the set of interest.

Histograms of the pixel intensities of an image can be used directly to alter its contrast. When pixel values occupy only part of the available dynamic range of displayable intensities, they can be redistributed to use more of the range of available intensities and hence increase the visible contrast.

When most or all of the dynamic range is used, but relatively large numbers of intensity levels are occupied by relatively few pixels, histogram flattening can be invoked. Typically, the inverse of the CDF is used to redistribute the pixel values.

When using the above techniques, care must be taken to avoid reducing the contrast rather than increasing it. For example, heavily occupied (i.e. background) intensity levels tend to cause excessive merging of the surrounding levels.

---

[10] Part of the work of this project was to write the program *enhance*. It provides a pleasant and powerful interactive environment for the implementation of and experimentation with filtering functions. See Chapter VI-D for details.

This can be overcome by imposing a ceiling on the histogram bins for the purpose of calculating the CDF. Another way is to measure the CDF of an area with a contrast range representative of the boundary conditions one is interested in.

These manipulations can be performed over regions of an image and can be restricted to affect only part of the dynamic range. Locations of peaks and valleys of the histogram of a region can be used to dynamically calculate spatially variant thresholds for other operations. Experimentation is usually required to determine how best to use histograms on a given image.[11]

False Colouring

Contrasting colours are a more powerful visual cue than are monochrome intensities. While enhanced contrast can reveal subtle details which might otherwise remain unseen in the original image, differentiation of pixel values using colours can result in a more dramatic visual effect. It is not necessary that specific colours be assigned to particular intensities or CT density values. Experimentally chosen colour combinations can yield good results.[12]

---

[11] The program *enhance*, written as part of the work of this project, includes a range of interactive histogram operations on images. See Chapter VI-D for details.

[12] The program *colours*, written as part of this work, provides a flexible interactive environment for selecting any 256 of the possible 2^{24} shades. See Chapter VI-C for details.

C. Automatic Contouring

Detecting and tracing the edges of regions of similar density yields a set of contours for each CT slice. In practice, this process is complicated by image noise and variations in image contrast which, though objectively small, may be subjectively significant.

The algorithm selected was judged to be the most promising found in the recent literature [19]. The authors were aware of problems with other methods and combined a number of techniques to attempt to overcome them. Several modifications were made to the original algorithm, but the basic strategy remains the same:

An edge magnitude image and an edge direction image are produced by a high-pass convolution operation from the original image which is left undisturbed and may be referenced by later stages of the algorithm. Each edge image pixel is then examined and classified as a superior-, an inferior-, or a non-boundary point based on the magnitudes of its 8 nearest neighbours.

If the image contrast is low, the edge magnitude image may be logarithmically scaled prior to the classification step[13]. The edge magnitude image is then scanned, pixel by pixel, line by line, to find a suitable point from which to start tracing a contour.

---

[13] This effectively increases the contrast but avoids introducing breaks in edge continuity such as those often caused by threshholding.

Tracing proceeds from one point to the next in a direction perpendicular to the gradient[14] at that point until a termination condition is reached (such as intersecting a previously-traced contour or the image boundary or simply stopping in a low contrast area). Tracing then proceeds in the opposite direction[15] from the same starting point until it also terminates. A forward and a reverse trace produce arrays of point-to-point coordinates which are retained in memory. The scanning process is resumed in search of other trace starting points.

At the end of a complete scan the contours are examined and classified. The non-closed contours invoke filtering of the original image in the rectangular regions formed by their endpoints. Subsequent re-scans and re-traces attempt to resolve these 'loose ends'. After a specified number of iterations, the contour coordinates are written to a file.

Modifications made to the algorithm include:

1. Replacing the Prewitt mask edge detector templates with Sobel masks having a variable parameter 'K' which was experimentally optimized to a value of 5/2,[16]
2. Removing image pixels belonging to previously traced contours and/or the 8 nearest neighbour pixels from consideration by the algorithm thereby allowing lower-contrast edges to be identified after the high-contrast edges have been extracted, and

---

[14] using the right-hand rule
[15] using the left-hand rule
[16] See Appendix 2 for the forms of Prewitt and Sobel masks.

3. Implementing a thinning operation which eliminates all coordinates except the endpoints of horizontal, vertical, and diagonal line segments during the output of contours.

D. Manual Intervention

The automatically-traced contours include contour fragments and artifacts along with the complete, closed curves. The artifacts should be removed and the fragments linked together and/or extended to form closed curves. Any of the curves judged to deviate from their optimum locations should be modified or replaced with manually entered curves.[17]

The auto-traced contours consist of adjacent pixels and thus total many hundreds of coordinate pairs per slice. To reduce the data to manageable levels, a thinning operation should be performed in which all points lying within a specifiable curvature are removed and replaced with a line segment Undue degradation to the shape of the curves should be avoided. Greater contour detail may have to be retained where required, by small contour radii for example.

When a stereotactic frame is attached to a patient's head during the CT scanning, the frame's locator assemblies appear as nine dots in an hexagonal pattern. To provide for the registration of all slices to a common framework and to

---

[17] The program *premos*, written as part of the work of this project, is an interactive contour editor. See Chapter VI-E for details.

a stereotactic coordinate system, the locations of these locator pins must be manually entered, in a consistent order,[18] for each slice [14]. The displacement along the z-axis which appears on each slice is entered as well and is currently used for model generation and display.[19]

After each set of contours is verified against its original CT slice image, the inter-slice relationship of the entire set of curves must be examined and possibly modified. Curves corresponding to the same structural features are gathered together as parts of the model and labelled accordingly. The data is then written to a file.

E. Inter-slice Registration

Any misalignment in the positions of the stereotactic locator pins from one slice to the next shows that registration errors have occurred. These errors should be removed by shifting all the contours of each slice by the amount of misalignment which has occurred.

A computer program which does this automatically is required.[20]

---

[18] See Plate 14.

[19] Any subsequent coordinate computation must be referred back to the frame pins rather than the (x,y,z) of the model coordinate system.

[20] The program *registrate*, described in Chapter VI-F was written to meet this requirement.

F. Triangulation and Display

A three-dimensional surface lying on the set of contours obtained through the previous steps must be generated. This data will be the final geometric description of the model and must be organized in such a way that it can be subjected to a full range of graphical display techniques including:

1. Viewing from any vantage point,
2. Surface shading and rendering,
3. Differentiation of various model parts using intensity and colour,
4. Withholding various model parts or making them transparent thereby revealing parts which would otherwise be obscured.

V. System Hardware and Software

A. System Hardware

Computer

The Logical Microcomputer Company (LMC) Megamicro System is based on the National Semiconductor NSC16000/32000 microprocessor chip set (8MHz) and includes:

- Demand-paged virtual memory hardware,
- Hardware floating point processor,
- 2.5 Mbyte 150 nanosecond RAM,
- 32 Mbyte Quantum fast 5 1/4 in. hard disk drive,
- 36 Mbyte Atasi fast 5 1/4 in. hard disk drive[21],
- 800 Kbyte (unique-format) floppy disk drive, and,
- 8 RS-232 serial communications ports.

Graphics Output

The graphics boards used are the Imaging Technology Inc. (ITC) IP-512 series including an FB-512 frame buffer, and an AP-512 analog processor (A/D & D/A). The ALU-512 math processor performs image convolutions at high speed but requires at least two FB-512 frame buffers and hence could not be used.

These boards drive a Barco GD 233 RGB colour monitor and obtain the necessary sync signal and video input from an Hitachi KP-120U video camera.

---

[21] no longer functional

Graphics Input Device

It was intended that a tactile hardware input device such as a graphics tablet, trackball, mouse or joystick would provide user interaction with the system. As none was available until the end of this project, a library of cursor functions and *cursor*, an operating system utility which move a crosshair cursor around on the screen in response to the terminal's arrow keys were written.

B. System Software and Support

Operating System

The system was first received with the *Unity*[22] operating system complete with a selection of bugs and generally poor performance. A major problem was the lack of a way to directly access physical memory from an application program.

Later, the manufacturer advised that the systems were being retro-fitted with *Genix*[23] and an 8 MHz CPU to replace the 6 MHz chip. System performance was much improved by these changes. The new bugs are relatively minor and fleeting. *Vspy()*, a system function call unique to *Genix*, provides the required access to the memory-mapped graphics boards.

---

[22] Human Computing Resource's version of Unix for the NSC16000 series microprocessor

[23] National Semiconductors' version of Berkeley 4.1 Unix.

Remaining bugs in the system include:

1. *Genix's* inability to consistently protect itself from user program address space violations.
2. Spurious console tty device lockup from which only a system re-boot gains an escape.
3. Unreliable serial line communication at baud rates over 1200, when more than 256 characters are transferred at a time.
4. Use of floating point math in C programs is sometimes straightforward and sometimes extremely difficult to comprehend. Manually scaled integer fixed-point was used where possible. Where floating point was necessary, perseverance and some interesting tricks eventually yielded runnable code.
5. The memory allocator *calloc()* does not always zero the allocated storage.
6. FORTRAN - C interaction does not conform to the documentation[24] and required a lot of effort to sort out. Functions written in C can be called from FORTRAN only if the subroutines which call them were compiled with *f77 -e ...*, and if these subroutines are argumentless. Access to the operating system from FORTRAN is limited to reading the command line arguments, file I/O, and calling argumentless system functions.
7. Debugging facilities *ddt* and *dbg* are clumsy to use and

---

[24] The documentation was never received from the manufacturer, but from another LMC user.

inadequate, both for C and for FORTRAN. In certain situations however, they can yield important clues and thus should not be avoided altogether.

LMC Installation

The Company did the bare minimum work in installing the operating system and in providing utilities. The disk formatter *formatit* worked but the procedure for formatting floppies was tedious and error-prone. User-unfriendliness also pervaded the backup utility *cp* which, if used as documented, would have yielded a mangled disk image upon reloading.[25] The other utilities, *cat*, *down*, *icheck*, *ls*, *memtest*, were either non-functional or useless.

The source code for the above routines was semi-present, but initially would not compile. After the needed pieces of code were found, collected, and modified, the above utilities could all be re-generated. Subsequently, *formatthem* and *cpthem*, user-friendly utilities were produced.[26]

These utilities are dependent on a particular set of hard disk configurations coded into the disk driver *dcu.c*. In the event of a catastrophic hard disk failure,[27] if the replacement disk does not have the same storage parameters

---

[25] Device blocks of 512 bytes were confused with system blocks of 1024 bytes within this utility.

[26] These utilities were the informal LMC User's Group which now serves as the sole source of support, LMC having folded during 1985.

[27] This has occurred in 100% of the Atasi disks shipped by LMC.

(number of heads and tracks) the utilities will not work correctly unless modified.

To provide for the ability to re-configure the utilities and system software, independent of the hard disks, a floppy disk called *mini.unix* which contains the essentials of the operating system was produced as part of this work.[28] The system can be booted up from this disk[29] and run using only the floppy drive as the main system storage device.

---

[28] See Appendix 3 for details.
[29] Use the monitor command 'bd', followed by 'dc(4,0)/vmunix'.

VI. Application Software

The main work of this project was to develop a body of computer programs which would perform the required functions outlined in Chapter IV. The strategy employed was five-fold:

1. Write and debug a library of graphics functions which would allow maximum and flexible use of the graphics boards by application programs.
2. Install and improve Movie.BYU; this included writing device drivers and modifying the main code.
3. Write the programs which would manipulate the images, extract the required contour data, and output it in a format compatible with Movie.BYU.
4. Scan the literature for techniques and algorithms used by others working on similar problems, select and implement the promising ones, evaluate them, and finally decide whether or not to incorporate them into the body of programs along with my own ideas.
5. Ultimately, make the system user friendly by streamlining the use of the various programs, files, and directories.

The first four of these were approached iteratively. Progress in one area sometimes facilitated progress in another, and sometimes required the upgrading of support functions and libraries. The fifth strategy, user friendliness was only partly achieved.

The resultant body of programs[30] consists of:

1. libraries of graphics and cursor functions which interface the graphics boards to the computer system,
2. graphics commands of various types available from the operating system,
3. *enhance*, an interactive image processing and automatic contour extraction program,
4. *premos*, an interactive contour line editor program,
5. *registrate*, a program which automatically aligns a set of contour lines,
6. *mosaic*[31], an interactive 3D surface generator program, (The output format of *premos* is such that *mosaic* is used automatically rather than manually.) and
7. *display*[32], the interactive graphics program used to display the geometric model generated by *mosaic*.

The structure of the resultant body of programs is illustrated by the diagram in Figure 2. Figures 3 through 12 show the structure of the individual application programs in detail. The format used in these diagrams is explained in Figure 1.

A high level of overall user-friendliness was not achieved. The command syntax of *enhance* and *premos* was made easy to learn and use, but the procedures needed to produce a view of a model generated from a set of CT slices requires greater attention to detail than should be the case. The

---

[30] written by the author except as noted
[31] part of Movie.BYU
[32] part of Movie.BYU

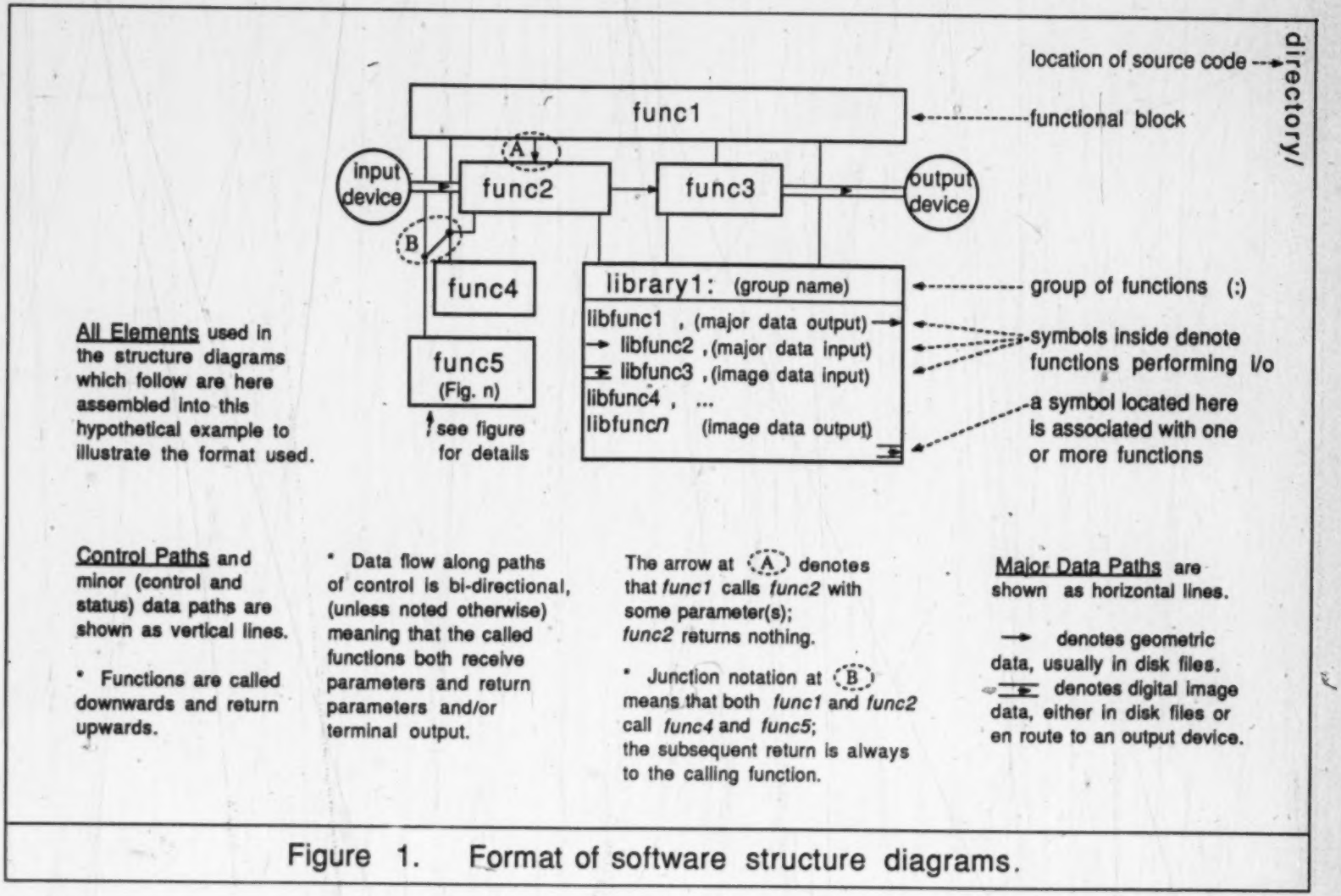
location of source code
directory/
func1
functional block
A
input device
func2
func3
output device
B
func4
library1: (group name)
group of functions (:)
libfunc1 , (major data output)
libfunc2 , (major data input)
libfunc3 , (image data input)
libfunc4 , ...
libfuncn (image data output)
symbols inside denote functions performing I/o
a symbol located here is associated with one or more functions
func5
(Fig. n)
see figure for details
All Elements used in the structure diagrams which follow are here assembled into this hypothetical example to illustrate the format used.
Control Paths and minor (control and status) data paths are shown as vertical lines.
* Functions are called downwards and return upwards.
* Data flow along paths of control is bi-directional, (unless noted otherwise) meaning that the called functions both receive parameters and return parameters and/or terminal output.
The arrow at A denotes that func1 calls func2 with some parameter(s); func2 returns nothing.
* Junction notation at B means that both func1 and func2 call func4 and func5; the subsequent return is always to the calling function.
Major Data Paths are shown as horizontal lines.
denotes geometric data, usually in disk files.
denotes digital image data, either in disk files or en route to an output device.

Figure 1. Format of software structure diagrams.

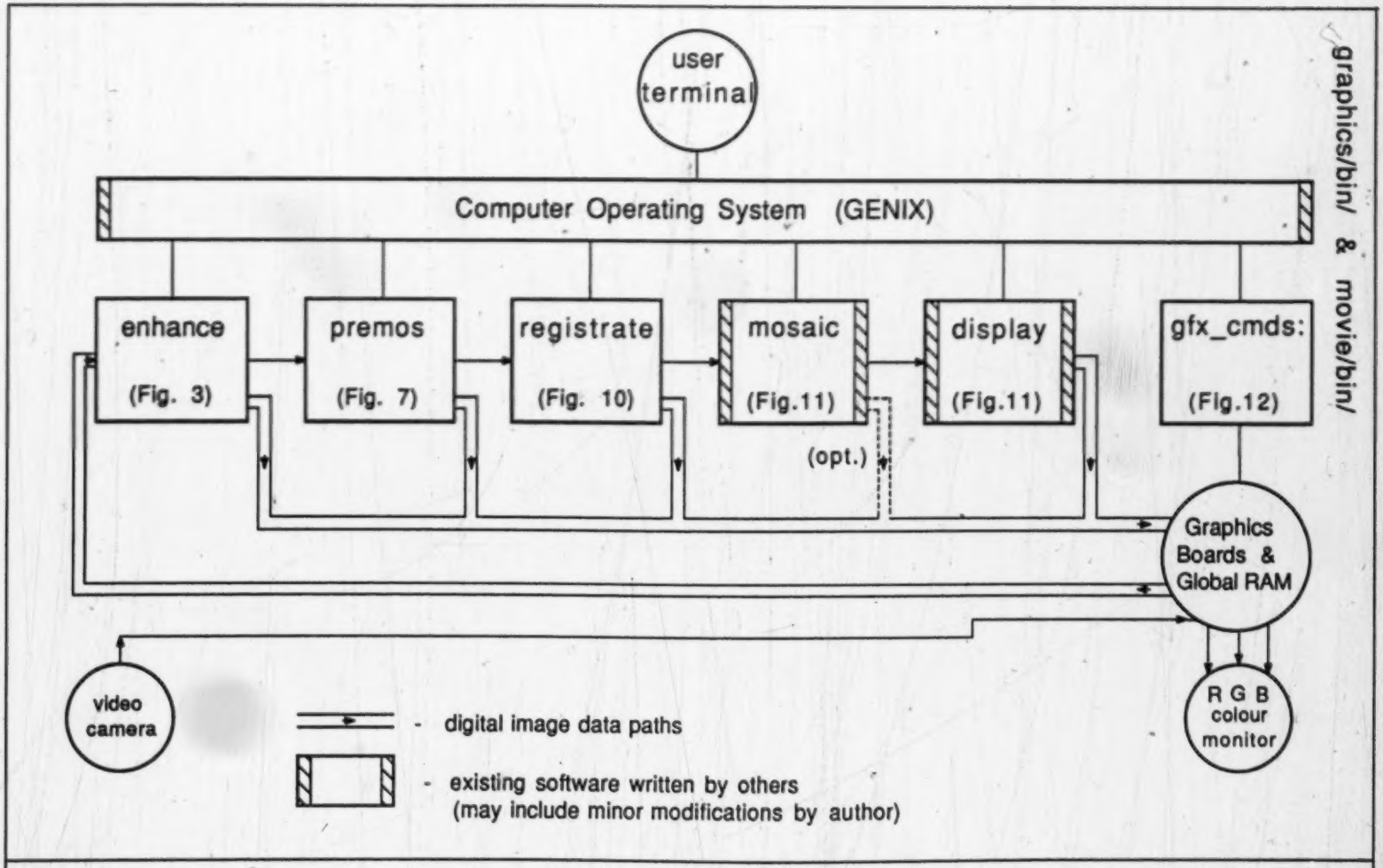

Figure 2. Structure of reconstruction system software.

Figure 3. Structure of 'enhance'.

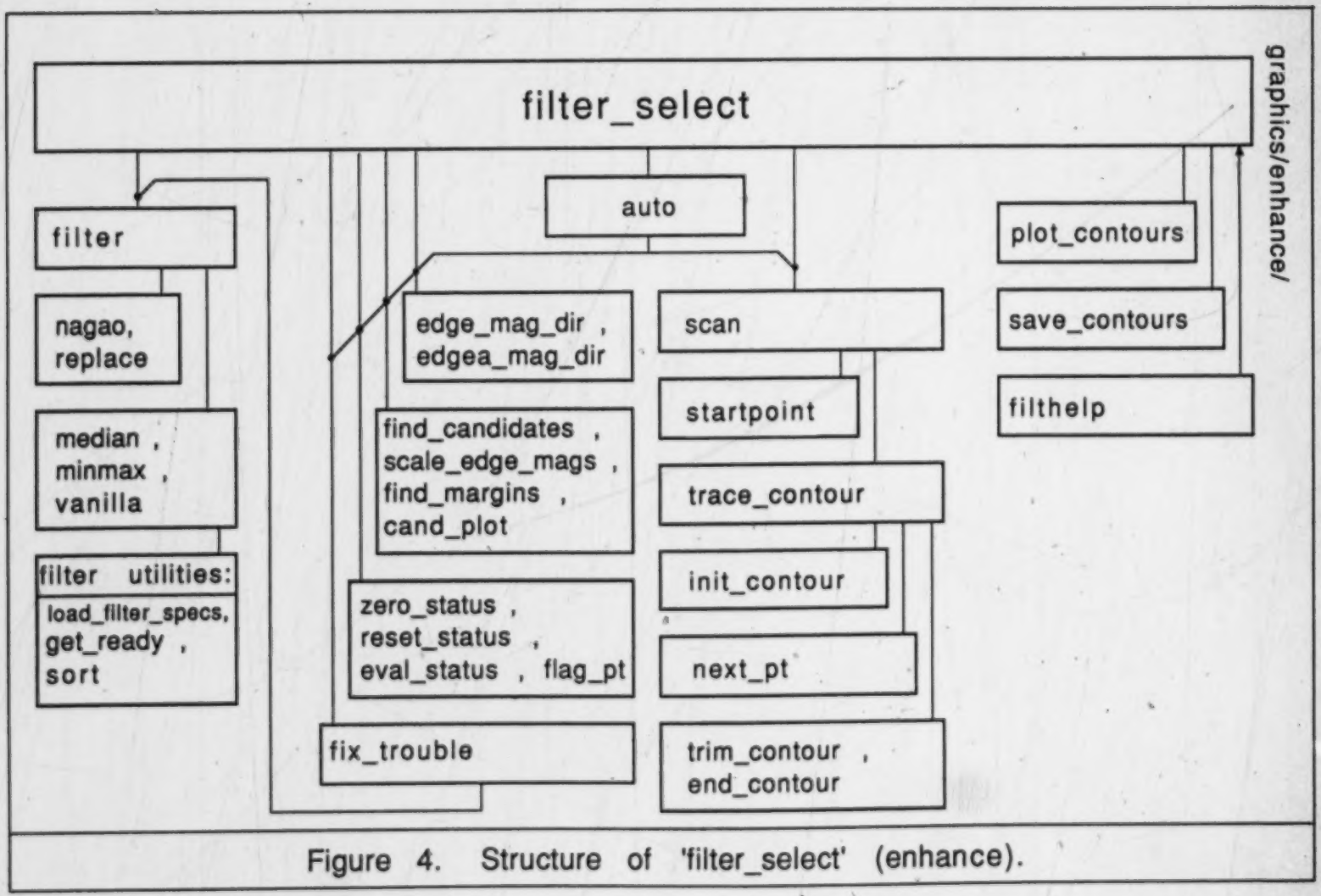

Figure 4. Structure of 'filter_select' (enhance).

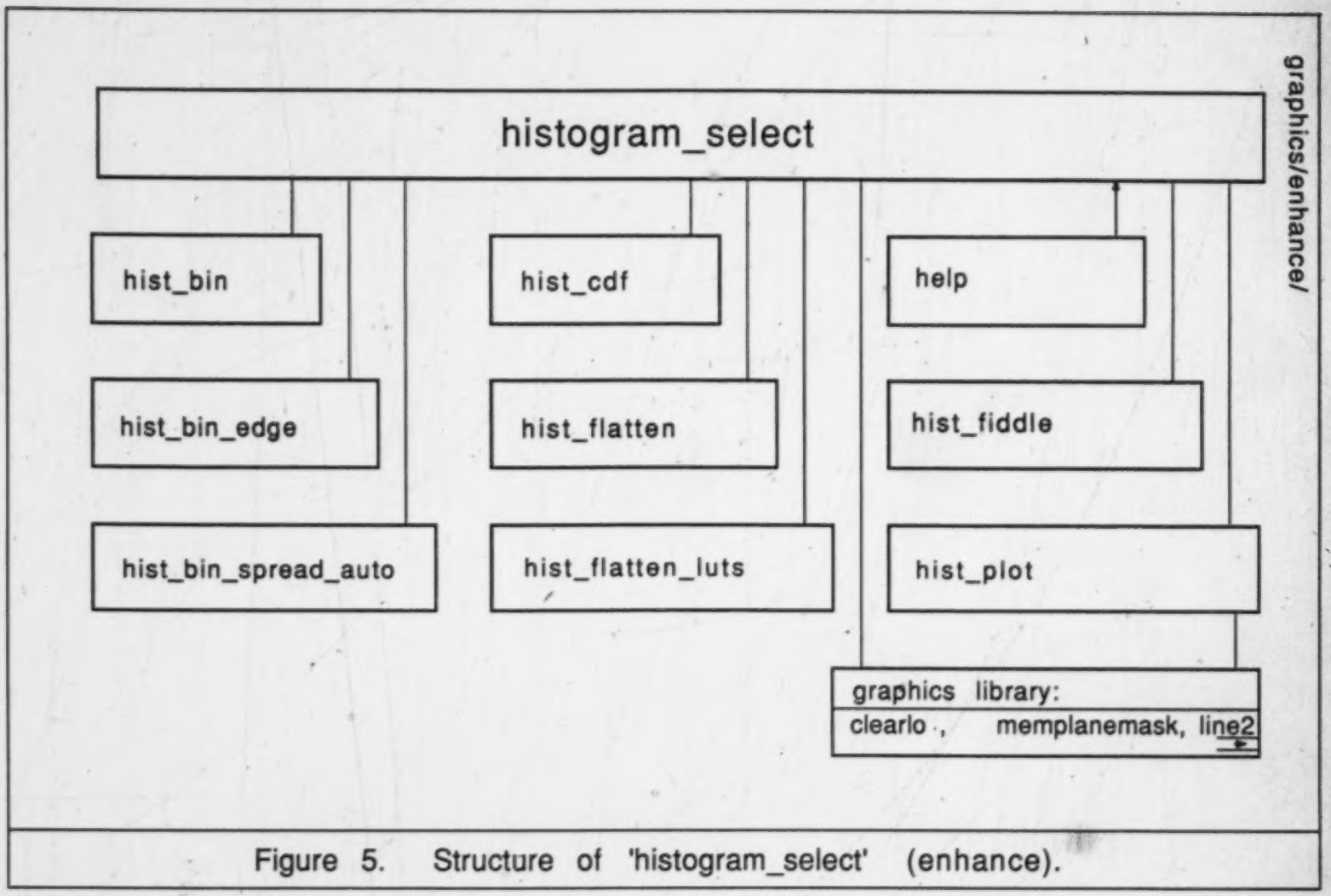

Figure 5. Structure of 'histogram_select' (enhance).

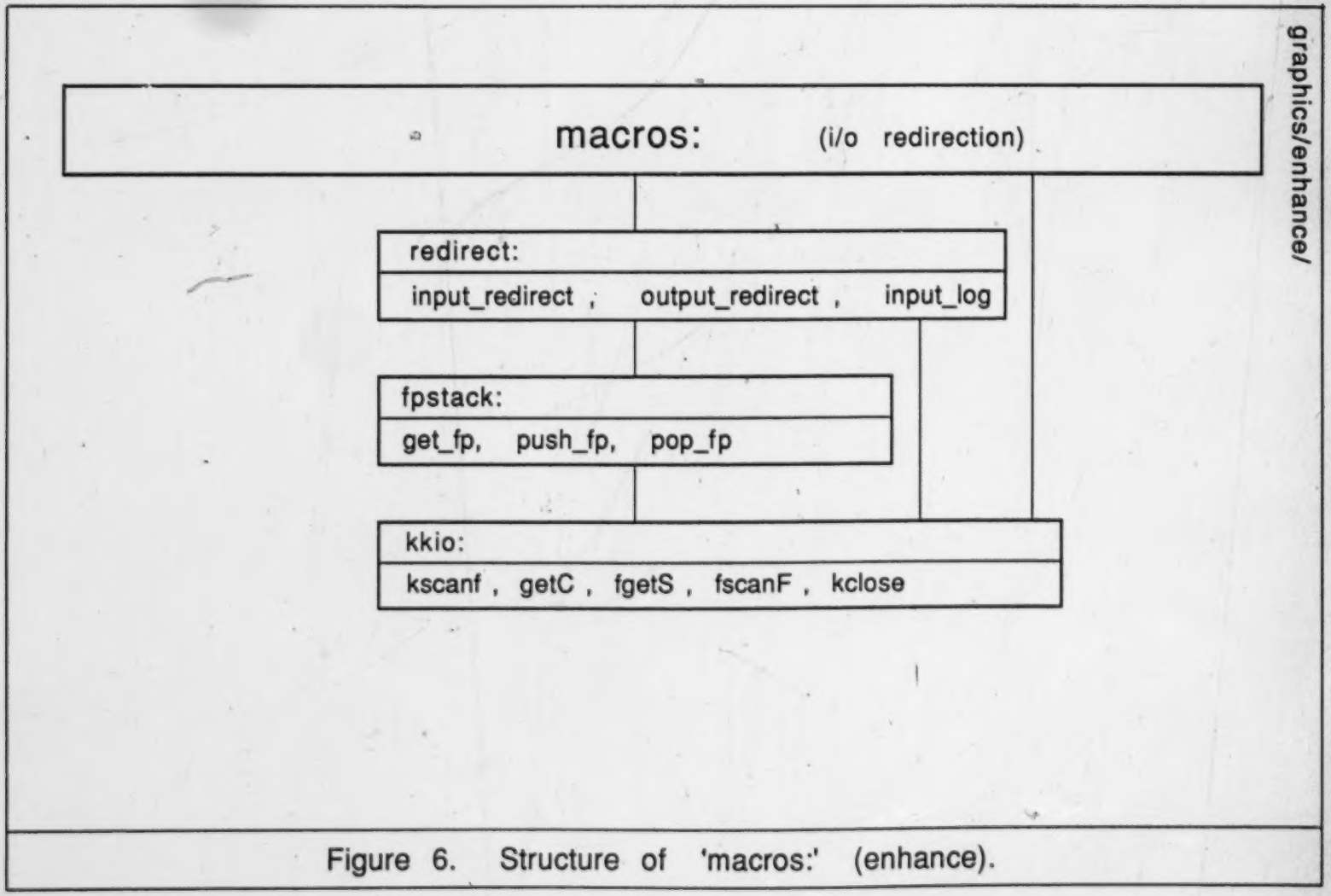

Figure 6. Structure of 'macros:' (enhance).

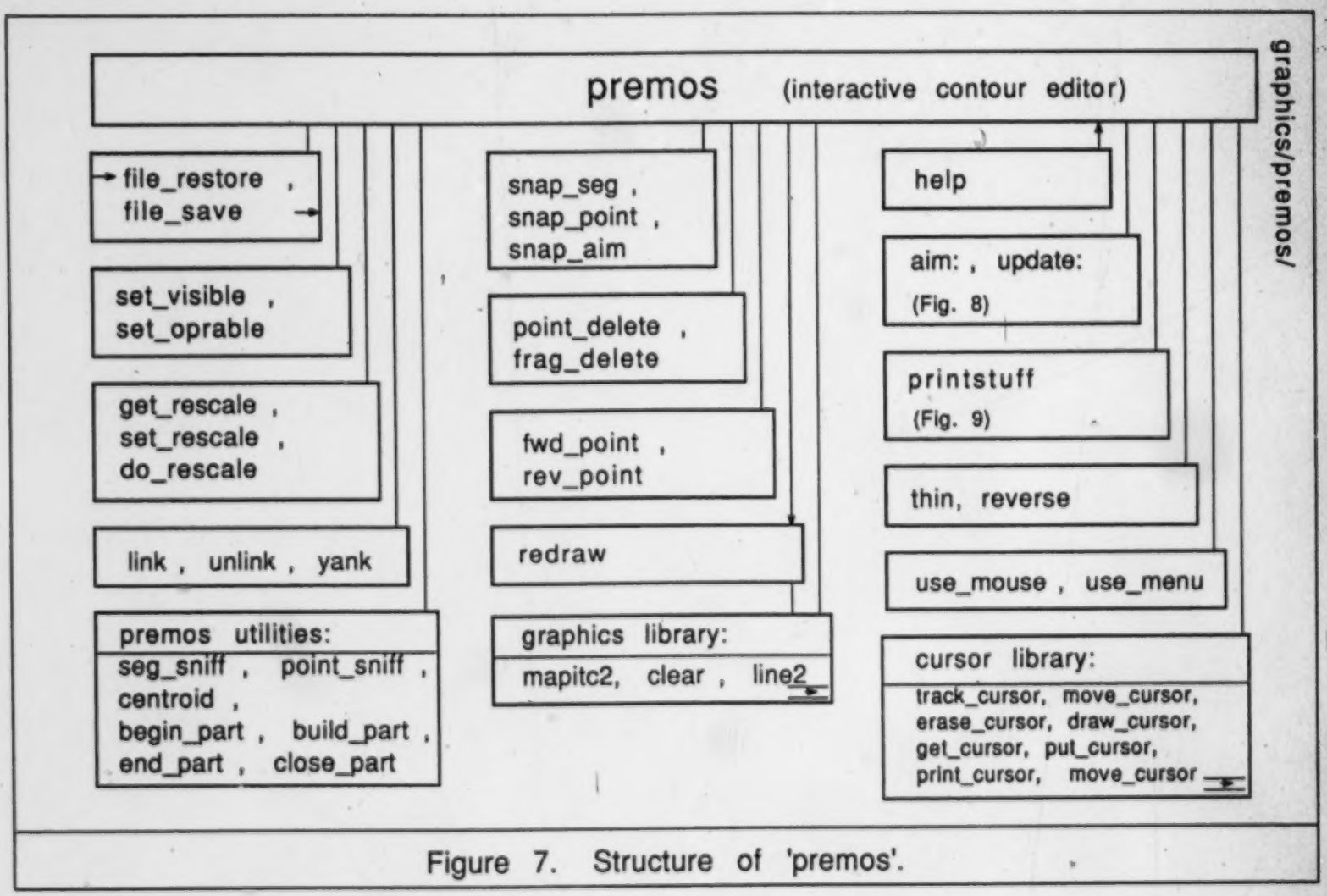

Figure 7. Structure of 'premos'.

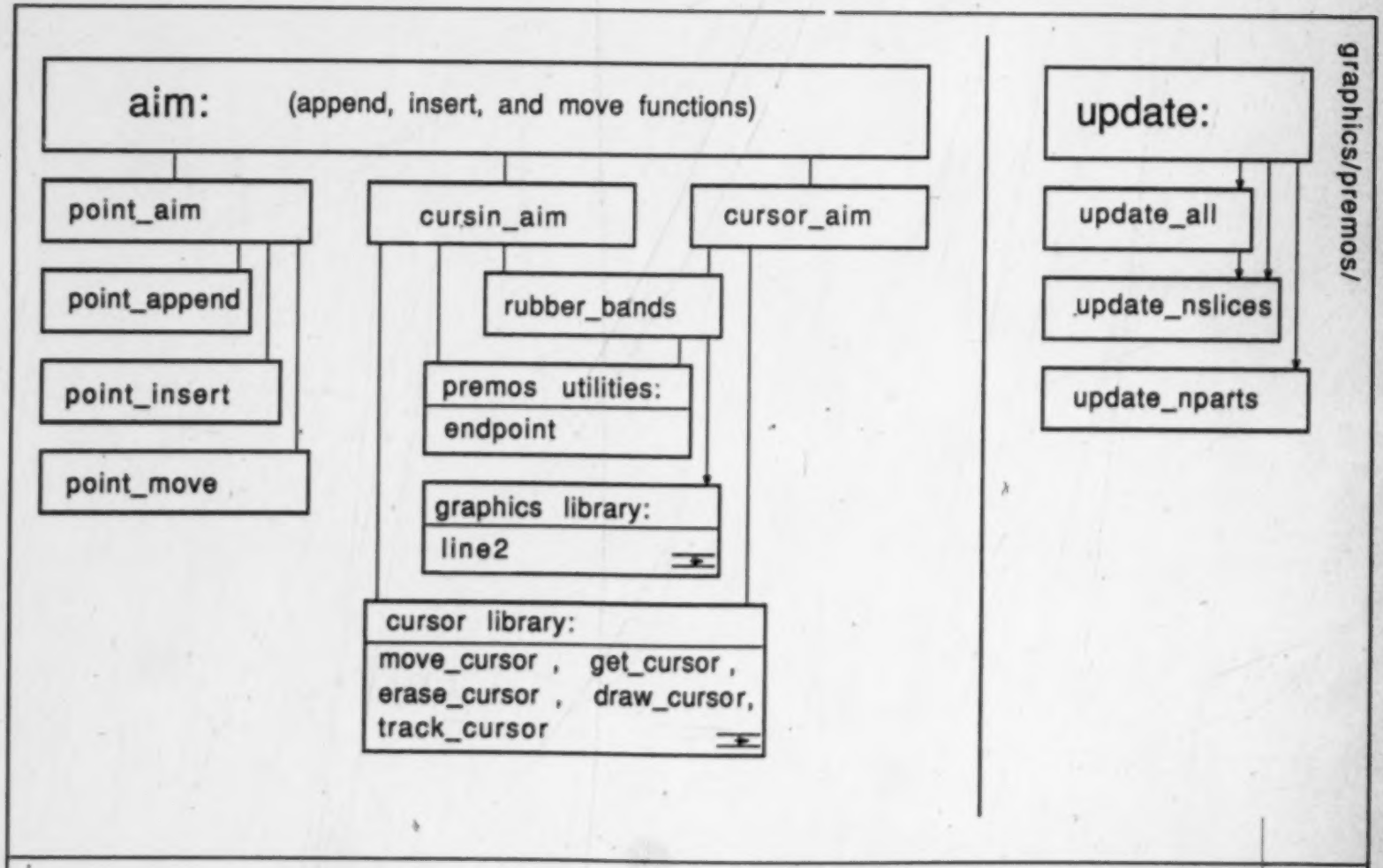

Figure 8. Structure of 'aim:' & 'update:' (premos).

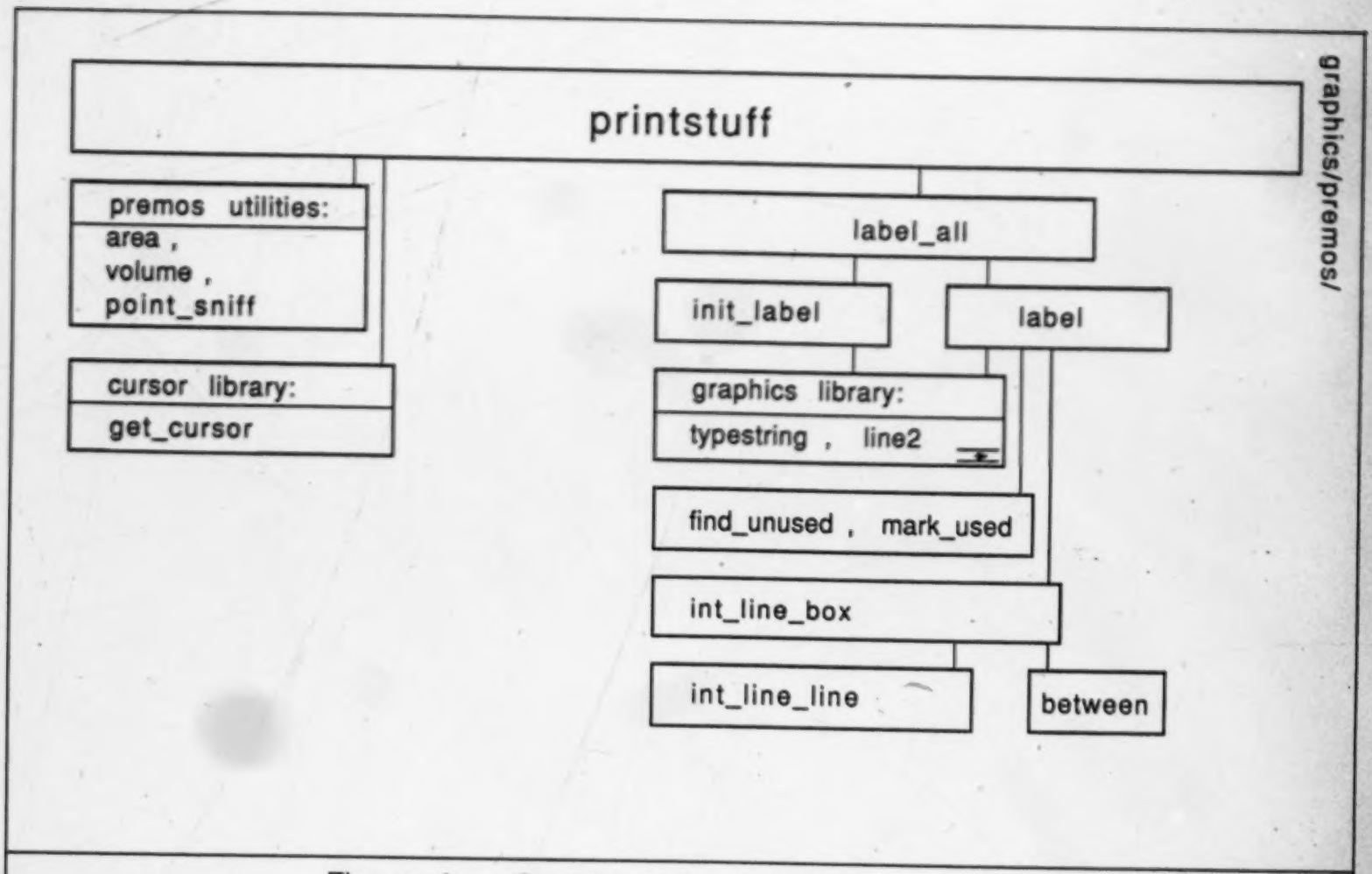

Figure 9. Structure of 'printstuff' (premos).

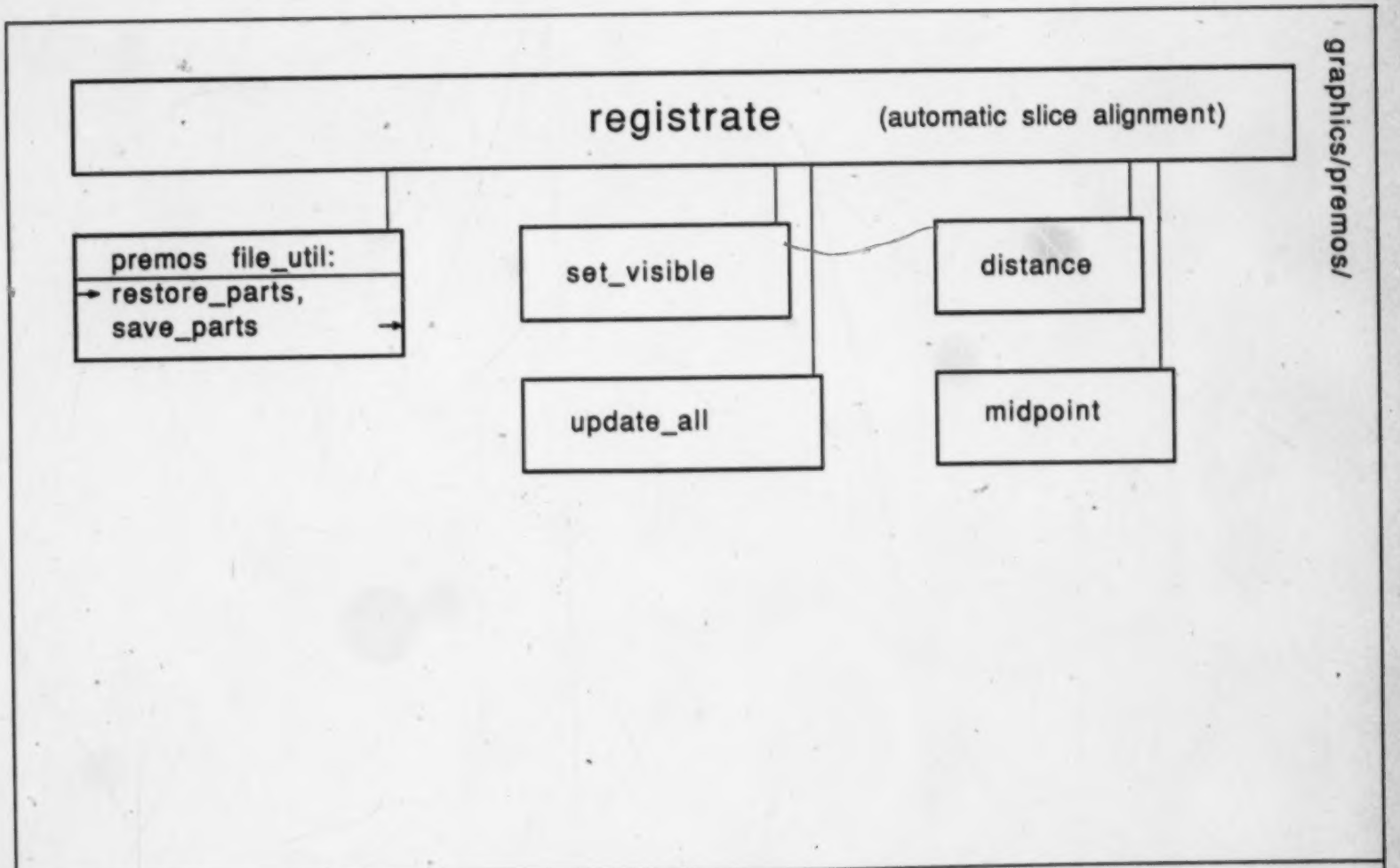

Figure 10. Structure of 'registrate'.

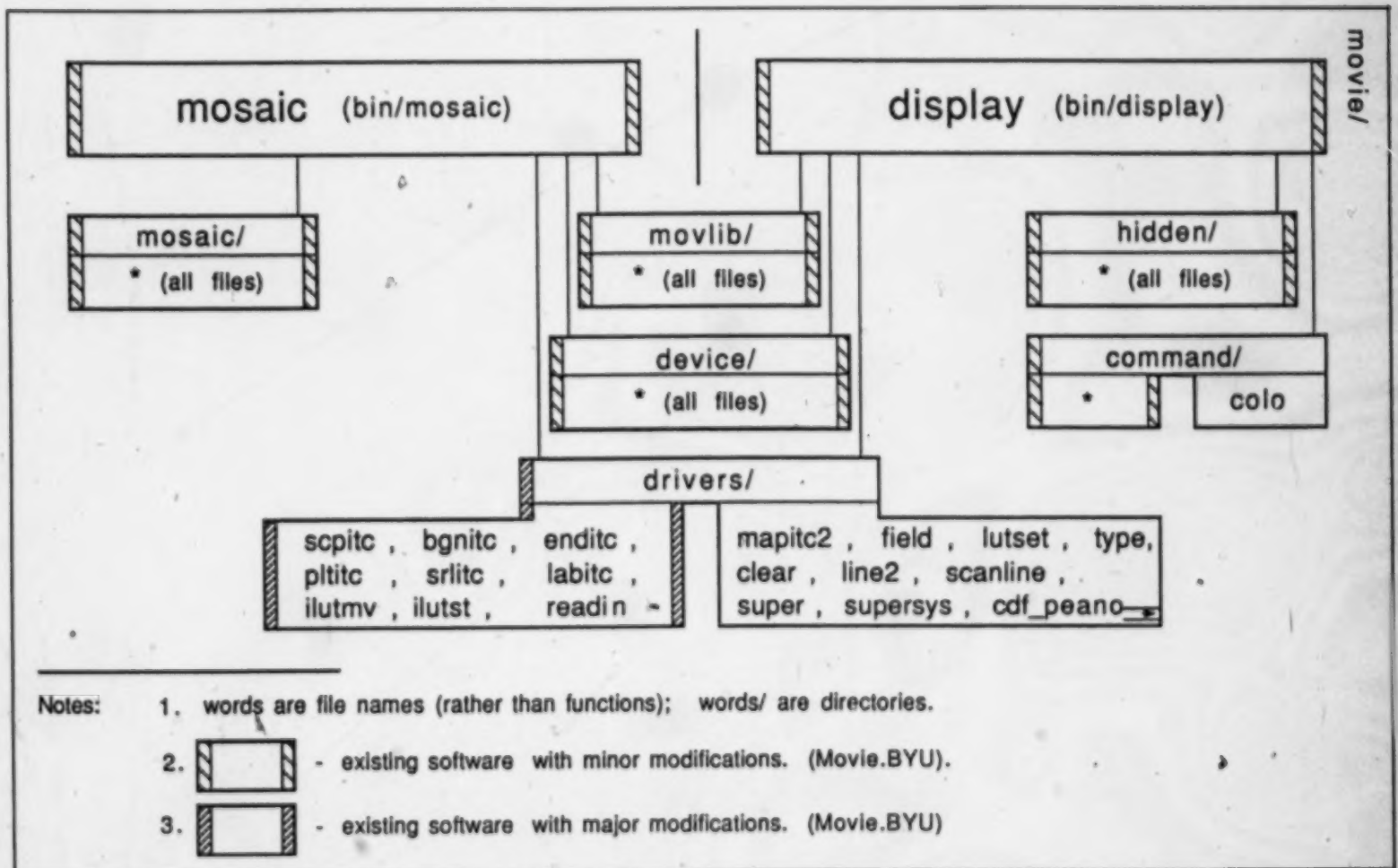

Figure 11. Structure of 'mosaic' & 'display' (Movie.BYU).

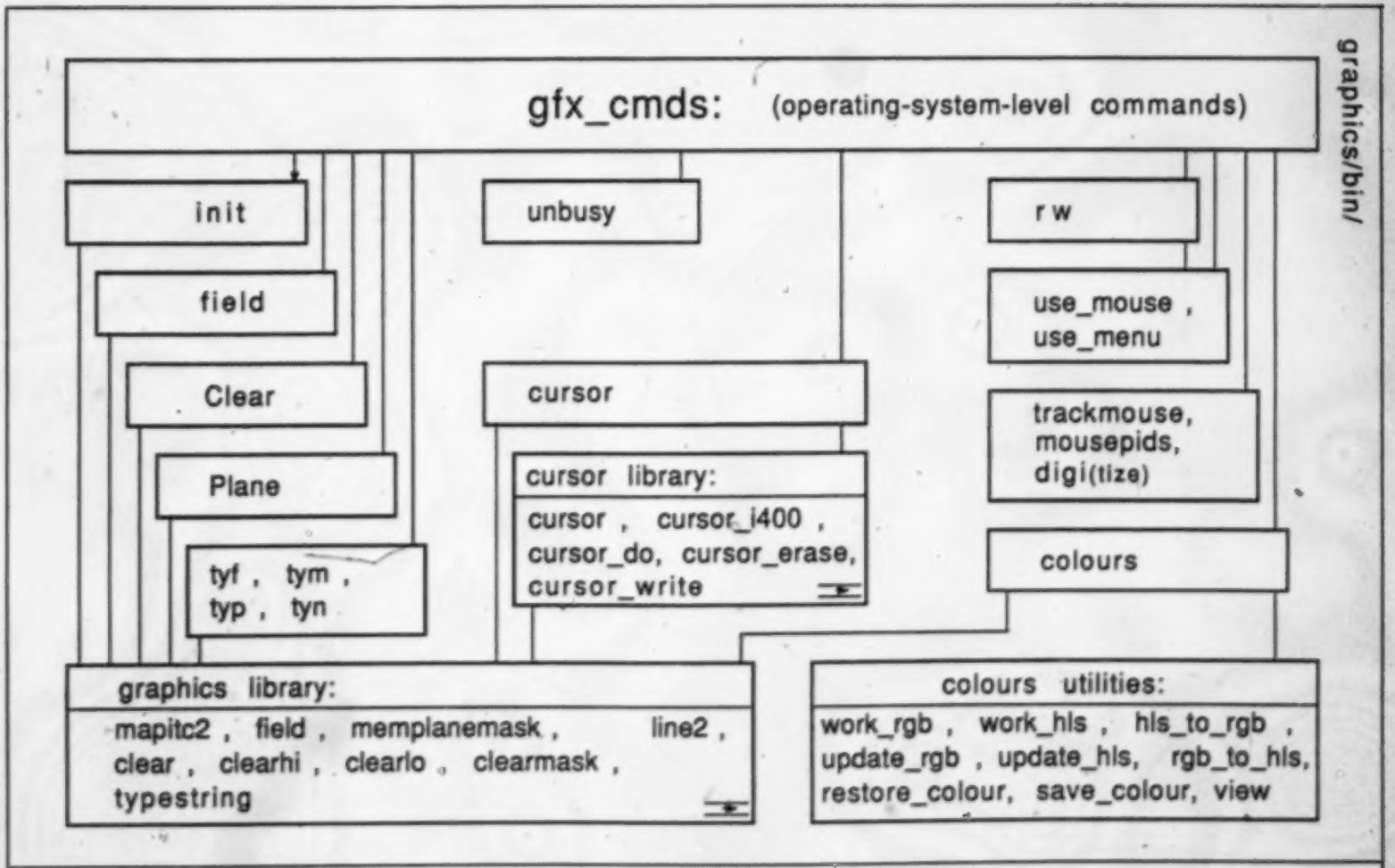

Figure 12. Structure of graphics commands 'gfx_cmds'.

operator must keep in mind the various programs and their associated data files; there is no automatic facility to guide the process or to deal with the intermediate files/operations for the user.

A. Graphics Boards Utilities

The functions *mapitc()*, *mapitc2()*, *field()*, *field_param()*, *memplanemask()*, *clear()*, *clearhi()*, *clearlo()*, *clearmask()*, *acquire()*, *grab()*, *line()*, *line2()*, *line3()*, *save()*, *restore()*, *msave()*, *mrestore()*, *operate()*, *subtract()*, *duplicate()* comprise the main body of device-dependent interface software.

The ITC graphics boards are controlled via registers[33] as per their technical manual [28].

Only indirect access to the image memory is available. Random pixel access requires the pixel coordinates to be written into the *xpos* and *ypos* registers, followed by an access on the *pixel* register. Raster pixel access, using the auto-increment feature, reduces the *xpos* and *ypos* register accesses to once each per raster line.

Interactive performance is boosted by maintaining a copy of the frame buffer in virtual memory. Most image operations are performed on this virtual copy which is sent to the frame buffer only on command.[34]

---

[33] See the include file *itc.h* for details.
[34] The access cycle time of the ITC boards is on the order of 1 us as compared with .4 us for the system RAM.

mapitc(), mapitc2()

One of these functions must be called in an application program before any calls to the following utilities. They establish a connection between the graphics boards and the current program. The ITC boards are seen by the system as two sets of registers memory-mapped into the physical address space at address ITC_Graphics (0x0c0e000). Access to this physical address is obtained via *vspy()*, a GENIX system call which maps and locks a virtual page onto a physical page.[35] Subsequent accesses to that virtual page, part of the normal user memory space, actually access the physical page.

The function *mapitc2()* is an extension of *mapitc()* which allocates a number of pages of "non-volatile" system memory in RAM. This global storage may be used for communication between programs running either concurrently, or sequentially.

memplanemask(m)

This function sets the memory plane protection mask to m and returns the old value of the register.

It was experimentally determined that the lowest 2 bits do not contribute significantly to the appearance of a grey-scale image. In addition, the frame grabber only

---

[35] Modification to the system kernel (*/vmunix*) was required to allow access to this call by users other than the super-user. The structure *spytable* in file *param.c* was changed to include the required physical addresses and the kernel re-'*make*'-ed.

digitizes to 6 bits. Simple and independent interactions between grey-scale images and graphics and/or text are required. To this end, the upper 6 bitplanes are designated as grey-scale images and the lower 2 bitplanes as graphic/text overlays. This provides for 64-level (or -colour) images, and 3-colour overlays. This convention can of course be overruled when required.

Memory plane protection is a feature of the graphics boards and affects all pixel write accesses. By our convention, a value of 0xfc protects the grey scale image while allowing write-access to the overlay image; a value of 0x03 does the converse.

clear(c)

This function uses the ITC clear feature to set the currently visible pixels to the value c (subject to the memory plane mask register setting). To ensure reliable operation, this function repeats the hardware clear a few milliseconds after its first completion. This function determines whether its argument is an integer or a pointer to an integer (as in FORTRAN calls).

clearhi(c), clearlo(c), clearmask(c,m)

These functions call *memplanemask( )* with *0x03*, *0xfc*, *m*, respectively, call *clear(c)*, then restore the memplanemask. This clears the image, graphics, and m-defined bit planes respectively, to the value c.

grab(),acquire()

These functions implement the single and continuous frame-grabbing features of the ITC boards.

field(f)

The frame buffer can hold an image of 512x512x8 bits of which 512x480 can be displayed at one time using interlaced scanning. Our video camera does not perform interlaced scanning, and since it provides the sync signal for the ITC boards, only the even or the odd field of 512x240 could be displayed at any one time. The ITC boards lock onto the even or the odd field arbitrarily after an interruption in the sync signal (i.e. by switching the input source selector to a non-existent signal). Bit 5 of the status register *fbctrl* indicates which field is currently displayed.

The first version of this function allowed the selection of fields 'e,o,b'.[36] This allowed for image size and processing time to be halved compared with a full 512x512 image, and for two images to be interchanged instantaneously on the screen.

The half-resolution modes in both x and y, combined with the scroll and pan features, suggested a further halving in image size to 65 Kbytes. Four 'quadrants' named q, r, s, and t, were defined and are loosely referred to as fields. They may be swapped instantaneously, or viewed simultaneously. Plate 3 shows the 4 quadrants as viewed

---

[36] The 'b' field depends on a rather unreliable crystal.

from the 'o' field; 'q' is the upper left quadrant, 'r' the upper right, 's' the lower left, and 't' the lower right. The current version of this function allows the selection of fields 'e,o,b,q,r,s,t' which occupy 128,128,256,65,65,65, and 65 kbytes respectively.

field_params(field,o,xoff,yoff,xinc,yinc,xdef,ydef,Y1,Yu)

Various parameters such as offsets, default field dimensions and indexing increments associated with the current field are made available to applications through this call.

The arguments, excepting field, are all pointers to *int*:

o=1 if field is 'o', 0 otherwise;

xoff=256 if field is 'r'or't', 0 otherwise;

yoff=256 if field is 's'or't', 0 otherwise;

xinc=1 always; yinc=2 if field is 'e'or'o', 1 otherwise;

xdef=511 if field is 'e','o',or'b', 255 otherwise;

ydef=479 if field is 'e','o',or'b', 239 otherwise;

Y1 = Y1/yincr*yincr + odd; Yu = Yu/yincr*yincr + odd;

line(colour,x1,y1,x2,y2), line2(...), line3(...,colour2)

The function *line()* is an implementation of Bresenham's line-drawing algorithm [29]. If 0≤ **colour** ≤0x0ff, frame buffer pixels are replaced with **colour**. If 0x100≤ **colour** ≤0x1ff, frame buffer pixels are exclusive-or'ed with **colour**. This function does not take fields into account; a line drawn in field 'o' will also

appear in field 'e'.

The function *line2()* is an extension of *line()* with coordinates (x1,y1) & (x2,y2) offset to conform to the currently visible field. A line drawn in field 'o' will not appear in field 'e'.

The function *line3()* further extends *line2()* with the addition of linear colour variation from **colour** to **colour2** along the length of the line.

typestring(field,string,x,y,xmag,ymag)

Alphanumeric text specified by the character array **string** is displayed on the monitor by writing pixel patterns directly into **field** of the frame buffer at the location (*x,*y). Magnification (by pixel replication) is specified by ***xmag** and ***ymag**.

Four fonts are currently available:

object module	width	height	cloned character set
typestring.of	8	10	Infoton 400 fixed width
typestring.op	3-8	10	Infoton 400 proportional
typestring.on	2-7	10	Infoton 400 narrow
typestring.om	3-10	13	Apple Macintosh prop.[37]

Each font definition consists of two manually composed files. One file contains the character bitmaps in a format such as *data.fixd* (periods are the background pixels). The other file contains the array sizes in a format such as *data.fixd.h*. These are transformed using *Makefile* into an object module such as *data.of* which defines the static bitmap and character width arrays. The character generator

---

[37] Thanks to Ron Unrau for entering this font definition.

driver in *typestring.c*, is compiled to reflect the size of the bitmap array, and combined with the above *data.of* module using *ld -r*. The resultant object module resides in *typestring.of*.

The file *main.c* contains the hooks to call *typestring()* directly from the operating system. Executable modules *tyf,typ,tyn,tym* are currently available. When invoked, the character cursor is located at the last position of the graphics crosshair cursor. This position is a local 'home' position which may be reset by two successive presses of the 'home'-key.

save(field,filedescr,pack), restore(...)

Save() writes field of the frame buffer to the file specified by filedescr.
Restore() reads the contents of the file specified by filedescr into field of the frame buffer.

If the argument pack is 'p', the image data is encoded/decoded. This typically results in a 50% reduction in file size.

msave(field,filedescr,pack), mrestore(...)

Msave() writes field in virtual memory to the file specified by filedescr.
Mrestore() reads the contents of the file specified by filedescr into field in virtual memory.

If the argument **pack** is 'p', the image data is encoded/decoded. This typically results in a 50% reduction in file size.

These functions do not access the graphics boards and are thus suited to non-interactive applications.

get(fielditc,fieldmem), put(...)

Get() copies **fielditc** of the frame buffer into **fieldmem** in virtual memory.
Put() copies **fieldmem** in virtual memory into **fielditc** of the frame buffer.

For these functions only, upper case arguments replace the normal raster access pattern with a randomized random access pattern. The square grid & block sizes default to 6 & 6, but may be reset with *get([1-9],[1-9])*.

duplicate(fielddst,fieldsrc)

Duplicate() copies **fieldsrc** in virtual memory into **fielddst** also in virtual memory. An 'e' or 'o' field can be 'shrunk' into the 'q' field (by omitting alternate horizontal pixels), and conversely, the 't' field can be expanded (by pixel replication) into an 'e' or 'o' field. Any other combinations of non-conforming fields may yield

unexpected results.

subtract(fielddst,fieldsrc)

Subtract() leaves the absolute value of the pixel-by-pixel difference between **fieldsrc** and **fielddst** in the latter. For this function, lower case arguments refer to fields in virtual memory and upper case arguments refer to fields in the frame buffer. Subtracting fields of differing sizes may yield unpredicted results.

operate(fielddst,fieldsrc,operation,x1,y1,xu,yu)

Operate() performs the operation selected by the character string **operation** between two sub-regions of fields in either virtual memory or the frame buffer. Calling this function with **fielddst**='x' and **fieldsrc**='d'/'s' sets the sub-regions in the destination/source fields from the arguments x1,y1,xu,yu. The pixel-by-pixel results of the operation between **fielddst**, **fieldsrc**, and some external constants are left in **fielddst**. For this function, lower case arguments refer to fields in virtual memory and upper case arguments refer to fields in the frame buffer. Operations on fields of differing sizes is supported but may yield unpredicted results. The operation '+' leaves dst+src in dst, '-' leaves |dst-src| in dst, '=' copies src to dst, and '=nnn' clears dst pixels to nnn. Consult the file *selfunc.c* to add new operators.

B. Cursor

These functions comprise a utility which moves a crosshair cursor around on the screen in response to the terminal's arrow keys:

cursor_do(c,x,y)	- does the actual work
cursor_i400(c,x,y)	- infoton400 interface to _do
stopwatch()	- key press interval timer
cursor(ipixel,x,y)	- interface to Genix

Key presses at slow rates (<1.5/s) yield pixel-by-pixel cursor movements on the screen. Progressively faster key events increase the cursor displacement/event thereby allowing full-screen traversal with a minimum of hits. The variable displacement rate strategy was inspired by previous use of HP test equipment which provided steps of maximum resolution at slow knob speeds yet allowed traversal of the full operating range via a few fast twists.

The file *generic.c* defines the following device-independent standard cursor functions:

get_cursor(x,y)	put_cursor(x,y)
move_cursor(x,y)	track_cursor(c)
draw_cursor()	erase_cursor()
print_cursor(fpo)	xprint_cursor(fpo)

get_screen_scale(xoffset,yoffset,xscale,yscale)

put_screen_scale(xoffset,yoffset,xscale,yscale)

The x and y positions as well as the screen pixel value are updated both in the ITC registers and in an area of global

non-volatile RAM.

C. Colours

Colours is an interactive program which provides a pleasant environment for manipulating the colours in the look up tables (LUTs) of the graphics boards. Its syntax is similar to that of the Unix/Genix text editor *vi* in that many commands are single letters which are executed immediately upon being typed.

The LUT values cannot be read back by the system, so therefore a floating point image of the LUTs is retained in global non-volatile storage and written to the graphics board as required. Manipulations are supported for both the RGB (red, green, blue) and the HLS (hue, lightness, saturation) colour spaces [29].

A colour is selected for modification either by direct numeric entry, or by pointing to it with the cursor. When do_all is clear, colours, beginning with the selected colour, in increments of 256/ncolours, are modified and assume the RGB values of the selected colour. In the {default} case of ncolours=256, only the selected colour is modified. When ncolours=4, as is the case for the greyscale with graphics/text overlay, up to 16 colours are modified at a time.

When do_all is set, by the 'A' command, the entire lutset is modified incrementally, colour by colour.

File operations are all done to the directory *maps* in the working directory, or else to the directory */usr1/graphics/colours/maps*. Colours with R=G=B=0 are not output to a file. This allows overlays to be created which can be read in on top of existing colour LUT setups.

Command Summary

A	Set do_all - for operations on the entire lutset.				
a	Clear do_all - for operations on single rgb values of the lutset.				
n	Set ncolours. {256}				
N	Set lutset#. {1}				
I	Set MaxIntens. {255}				
u	Update the graphics board LUTs. from the LUT image in memory.				
v	View the pixel, R, G, B, H, L, S values.				
y	Yank the rgb's from colour @ current cursor.				
p	Put the rgb's to colour @ current cursor.				
z	Zero all rgb.				
i	Set increment used by the following 16 commands.				
t	translate	+=incr	T	translate	-=incr
r	red	+=incr	R	red	-=incr
g	green	+=incr	G	green	-=incr
b	blue	+=incr	B	blue	-=incr
w	white	+=incr	W	white	-=incr
h	hue	+=incr	H	hue	-=incr
l	lightness	+=incr	L	lightness	-=incr
s	saturation	+=incr	S	saturation	-=incr
=	direct entry of last-twiddled parameter				
fs	Colour save to file ...				
fr	Colour restore from file ...				
^V	Literal next character				

?	Help
!	Shell escape to operating system.
q	Quit

When 'colours arg1 arg2 arg3 ...' is invoked, the command line arguments are executed before commands are read from the input. For example, 'colours N3 fr greyscale fr ov.8.bold 0 w=0 fs fred q' will change the LUT set to 3, read a greyscale definition file, overlay it with 8 bold colours, set the colour of intensity level 0 to black, save the result in the file fred, and exit the program.

D. Enhance

Enhance is an interactive program which provides a pleasant environment for manipulating images. Its syntax is similar to that of the Unix/Genix text editor *vi* in that many commands are single characters which are executed immediately upon being typed.

Shell Command Summary

A '.' in the first 7 commands stands for a field specification character, one of ' vhbeoqrst'. A 'v' means the visible field, an 'h' the hidden field, and a ' ' is interpreted as 'v' or 'vv' depending on context.

h.	Histogram functions subshell.
F.	Filter functions subshell.
f.	Field select.

g..	Get screen > memory.
p..	Put screen < memory.
=..	Duplicate a field.
-..	Subtract 2 fields ('.' may also be 'VHBEOQRST').
@..	Operate on 2 fields ('.' may also be 'VHBEOQRST').
v	View current parameters: field, box-cursor values.
b	Set box-cursor numerically.
u	Set box-cursor upper corner from cursor.
l	Set box-cursor lower corner from cursor.
U	Cursor to upper corner of box-cursor.
L	Cursor to lower corner of box-cursor.
m	Move the current box-cursor.
?	Cursor coordinates.
I	Initialize visible screen.
c[hlm]	Clear [hi lo mask] [val].
e	Erase box-cursor.
d	Draw box-cursor.
sa/sp	Save to file ...
Re/Rp	Restore from file ...
re/rp	Restore from file ...
!	Shell escape to operating system.
Q	Quit.
H	Help menu displayed.

Macros

The following commands allow for the creation and use of macros which are files containing sequences of shell and

subshell commands. The input and output redirection syntax is borrowed from Movie.byu and the communications utility *cu*.

>[>]	Input logging to file; Macro creation.
<	Input redirect from file (may be nested 10 deep); Macro usage.
&>	Output redirect to file; - program output and echoed input.
(/)	Expand macros = on/off; while reading from a macro file, the (entire macro contents/just the macro name) are echoed on the output or the logging file.
#	Comment line; input line is ignored.

In the current implementation, any characters not accepted by the basic shell's command parser are passed on to the cursor tracking function. When a graphics input device is incorporated, unaccepted characters should sound a warning and be discarded; a separate function should update the system's crosshair coordinates.

When the following subshells are invoked, a line of input is read in and stored. For each word recognized, [optional parameters], if present, are scanned from this line and the appropriate static variables updated. All subshell functions are executed over the area of the image enclosed by the box-cursor. Their return codes are displayed in square brackets upon completion; an error condition (return code <0) causes the subshell to exit immediately.

Filtering Functions

Each word not recognized by the parser is assumed to be a filtering function file name. The directory */usr1/graphics/filter/filterspecs* is searched and if the file is found, its contents define a convolution kernel.[38]

edge [ek,es]	Sobel-based (K=ek/es) edge detector for auto contouring algorithm.
edgea [ek,es]	Absorption (K=ek/es) edge detector.
findcand [ct]	Find candidate edge points.
scale	Apply logarithmic scaling to edge magnitude image.
findmarg	Mark box-cursor boundaries on edge image.
resetstat	Delete all previously-traced contours and reset status array.
zerostat	Delete all unflagged previously-traced contours.
scan [min]	Initiate scanning/tracing process.
evalstat [ev]	Examine the status of traced contours. Contours with fewer than ev points are deleted.
fix [-ss]<rc>[rc]	Invokes filtering on endpoint rectangles of un-closed contours. -ss is the filename of the filter which defaults to "replace". If **evalstat** immediately preceeds, rc gives number of contours in need of fixing.
auto [i,ev,-ss]	Initiate complete auto-contouring operations. The number of iterations is denoted by i. -ss is the filter filename as in fix.
plot [b,e,t,m]	plot contours b(eginning) to e(nd) at m(agnification) The absolute value of t(hreshold) specifies the minimum number of coordinates for a contour to be

---

[38] Appendix 1 gives a summary of, and Appendix 2 the kernel definitions of the available filters.

plotted.

print [b,e,t,m]	print contours b(eginning) to e(nd) at m(agnification) The absolute value of t(hreshold) specifies the minimum number of coordinates for a contour to be printed. If t(hreshold) is <=0, only headers are printed.
save [sl,z]<betm>	The axial displacement of the current sl(ice) is specified by z. The parameters b,e,t,m last set by the print or plot commands are used by save.
zeropt	Set all unflagged points to 'untraced'.
flagpt	Flag all superior-boundary points
!flagpt	Unflag all points.
squeeze	Delete all un-closed contours, re-order the list.
flag	Flag all traced contour points.
!flag	Unflag all traced contour points.
maskpt	Select action to be taken in cases of ambiguous edge direction. Default action is to choose the direction randomly. Invoking this command causes the ambiguity to be resolved by using the edge direction of the previous pixel.
putemag	Put edge magnitude image into virtual buffer.
putmem2	Put alternate (filtered) buffer into the virtual buffer.
help,?	Display a help menu.
lf	Display available filtering function types.
ls	Display available filtering filenames.
cflag []	Set level of function tracing {0}
!g	Toggle the bell-on-completion flag.

Histogram Functions

An unrecognized word causes the help menu to be displayed. If the autoplot feature is on (the default), the graphics overlay is cleared and the histogram and CDF are plotted upon exiting this subshell.

nbins	Set the number of bins. {64}
bin	Binning - totalizes the number of pixels of value 0-255 into array of size nbins; the array index is scaled by 256/nbins.
bins	Binning_spread with 4:1 pixel value redistribution based on adjacent pixel values.
binsa	Hist_bin_spread_auto; see source code for details.
bine	Binning only of pixels near edges.
bineq	Binning of pixels near edges - quickly, without the implicit filtering operation.
cdf	Calculate the cumulative distribution function of the values currently in the bins.
flat	Flatten the image pixel distribution by replacing pixels with their inverse-cdf function values.
flatluts	Flatten the displayed pixel intensity values by modifiying the video lookup table according to the inverse-cdf function.
peaksall	Find all peaks and valleys in the bins.
Peaks [T,Ti,Tm]	Select significantly 'peaky' peaks; does peaksall first.
peaks [T,Ti,Tm]	Select significant, 'peaky' peaks.
peak [i,cum,dir]	Select first peak with at least i% of bin index, or cum% of total pixels; dir specifies minimum acceptable peak amplitude.
peakmagic	Select one optimum peak; see source code

for details.

psmooth [a]	Perform simple exponential smoothing on the peak values.
smooth [a]	Perform simple exponential smoothing on the bin values.
print	Print the values in the bins, and of the cdf.
Print	Print the values of the peaks and valleys of the bins.
plot	Plot the bins, cdf, peaks... properly scaled to the specified field.
Plot	Toggle auto-plot. {on}
min,max	Set the range of grey scale values used by the flattening function {0-255}
fiddle [f]	Impose a ceiling on the histogram bins so that any bin value greater than f is set to f. The pixel total is decremented by the difference to retain an accurate count.
cflag []	Set level of function tracing.
^g	Toggle the bell-on-completion flag.

E. Premos

Premos is an interactive graphics editor which facilitates manual intervention in the generation of a set of contours. Its syntax is similar to that of the Unix/Genix text editor *vi* in that many commands are single letters which are executed immediately upon being typed. The input and output files are in a *mosaic* command file format, hence its name.

Once read in from a file, or manually entered using the cursor, the data in memory consists of contours and contour

fragments organized by 'part' and by 'slice'. These curves can be displayed and modified either on a blank background or superimposed on the original greyscale or false-coloured CT image.

The 'visibility' of parts and slices is controlled by the command 'v', and their 'operability' by 'V'. Commands which search for existing data will only find a fragment whose part and slice are both visible. Commands which modify data will work only on a fragment whose part and slice are both operable.

Commands which require coordinate entry will 'rubber-band' the line segments from the current cursor location to the adjacent point(s) to show what is happening. A '.' will tell the command to accept the current cursor location as its current point. A carriage return terminates the command.

Command Summary

fs/fr	File save/restore which first makes all parts and slices visible and operable and resets any part and slice offsets to zero.
Fs/Fr	File save/restore which operates only on operable/visible parts and slices. The parts are offset by currentpart-1. The slices are offset by currentslice-1. This mode can be used to merge files or to save only part of the data.
p	Set currentpart and part label.
s	Set currentslice (z).
b	Begin currentpart & currentslice at the current cursor location. Points are added using '.' with the cursor in the desired position until an 'e' or 'c' terminates the command.

e	End part, increment currentpart.
c	Close part - add a segment from the last point to the first, increment currentpart.
a	Append @ endpoint of nearest operable curve.
i	Insert @ nearest operable point.
m	Move nearest operable point.
>	Fwd cursor to next point on the current contour.
#>	Fwd cursor to last point on the current contour.
<	Rev cursor to prev point on the current contour.
#<	Rev cursor to first point on the current contour.
*	Snap cursor to nearest point and update the current contour.
.	Use point at current cursor location.
, & x	Delete nearest operable point, or the last manually-entered point.
X	Delete a contour.
@	Move cursor to centroid of and print the area of the current contour.
?	Print the current cursor location.
l	Link two open loops. The resultant loop will retain the number of the loop first selected. A loop may be linked to itself to close it.
L	Unlink at current segment. If the segment is adjacent to an endpoint, it is simply deleted; otherwise, a new fragment is generated consisting of all the points from the segment to the last point. The new fragment will be placed into the first operable part available on that slice.
=	Duplicate the visible parts and slices into the operable parts and slices.
r	Reverse the point ordering of the current contour.
↑L	Redraw the screen.
↑K	Redraw the screen with crosses.

↑R Autoredraw toggle.

o Set offset&scale params.

%O Do offset&scale on operable parts and slices.

vp[0-9a-z*^] Set visible and operable parts.

Vp[0-9a-z*^] Set operable parts.

vs[0-9*^] Set visible and operable slices.

Vs[0-9*^] Set operable slices.

v* All parts & slices visible and operable.

V* All parts & slices operable.

t [minlen,maxlen,maxdiff]
Thin the contour. Any series of points lying within a specified curvature is replaced by the endpoints of an equivalent line segment.
This command prompts for the optional input of the parameters *minlen*, *maxlen*, *maxdiff*. A point is subject to deletion if the distance to the previous point is less than *minlen*, or if the difference between the euclidian distance and the distance along the original curve to the last-retained point is greater than *maxdiff*.
A point subject to deletion is retained if the resulting segment length would be greater than *maxlen*.

%z Clear operable parts & slices.

%S Shrink - parts are re-ordered with empty parts removed.

P[ps 1] Print information about the visible contours.
'Pp' will print the number, name, number of occupied slices, and volume of each visible part.
'Ps' will add to each of the above part headers the slice number, number of points, and area of each occupied contour of each visible slice.
'P ' will add to each of the above contour headers the number and the coordinates of each point of the contour.
'P1' will not print anything, but will label each part and each slice on the display.

! Shell escape to operating system.

q! Quit.

H Help menu is printed.

F. Registrate

The utility *registrate* operates on files in *premos* output format. The assumption is made that part #1 of each slice consists of the 9 points of the stereotactic frame digitized in conventional direction, beginning and ending at the left-most point as shown in Plate 14. For each slice, linear scale factors and translation constants in the x and y directions are calculated by fitting points #1, 2, 4, 5, 7, 8 to the geometry of the frame which is usually hexagonal.
The remaining points on the slice are then subjected to the same transformation.

The output file may be re-examined by *premos*, as seen in Plate 16, or passed on to *mosaic* for triangulation.

G. Movie.BYU

Movie.byu is a system of

> *"FORTRAN programs for the display and manipulation of data representing mathematical, architectural, and topological models whose geometry may be described in terms of panel and solid elements, or contour lines".*

written at Brigham Young University.[13]
The programs *mosaic* and *display* are installed and used in this system.

The Movie.byu source code was downloaded from a UofA Computing Science VAX using *uucp* via a serial line. The majority of it is device- and system-independent and was compiled without difficulty. The following changes were made to the device-independent source code in the installation process:

1. The variable OUTPUT, unit number for terminal communications, was changed from 1 to 0 thereby directing terminal output to *stdout* rather than to *stderr*.
2. The characters ",$" were added to FORMAT statements governing terminal input prompts to allow the cursor to remain on the same line as the prompt.
3. The subroutines COLO and PCOLOR were rewritten to provide colour information to the device-dependent screen-clearing and line-drawing functions. *Display*'s 'COLOur' command now allows input values in either of the ranges 0.01-1.0 and 2-255. A part number range followed by a return now prints the existing colour settings, leaving them unchanged.
4. The array size constants in the user-modifiable or 'variable'[39] DIMENSION statements were replaced with macro names which are defined in *Makefile* and resolved by the pre-processor *cpp* prior to compilation. Modifications to reflect model sizes and complexity can now be done quickly and accurately.

---

[39] This term is used in the source code and the manual [13].

Mosaic

The program *mosaic* generates three-dimensional surfaces lying on sets of contours by generating triangles between nearest points of adjacent pairs of contours. As a means of automating this step, files output by the program *premos* contain sequences of *mosaic* commands and data. These commands first 'MAKE' files of reformatted coordinates named *part*n, one for each 'PART' of the model. They then 'READ' these files, 'AUTO'-triangulate to generate the surfaces, and finally 'WRITE' the 3D model description (in terms of nodes and connectivities) into a file named *geom* which is destined for *display*.

Mosaic has a wider range of commands than those above and can be used interactively to gain more control over the triangulation process if necessary. For a complete description of all *mosaic* commands, the reader is referred to the "Mosaic User's Manual" [13]. Normally, however, the automatic procedure currently being used should be adequate.

Display

The 3D model can be viewed from any vantage point with a variety of attributes and physical characteristics using the program *display*. Parts can be differentiated by colour, be withheld from the display, be made transparent allowing obscured parts to become visible, and so on. The model may be rotated and translated to provide the desired viewing orientations. The viewer may also zoom-in or zoom-out,

select the degree of perspective distortion desired, or arbitrarily position light sources to provide pleasing and informative surface rendering.

The following *display* commands are the most important for this application. For a complete description of all *display* commands, the reader is referred to the "Display User's Manual" [13].

SCOP	Selects the desired output device with a choice of 4 modes and 2 resolutions. Must be used before the first DRAW or VIEW command.
DRAW	Draw the wire frame picture of the model defined by all the previous commands. Hidden line processing is done only if FAST was issued.
VIEW	Draw the picture, but invoke hidden line processing, and if applicable, surface rendering.
LIGH	Position the light source(s).
FIEL	Define the frustrum of vision and clipping planes. Decreasing the angle will increase the model size and decrease the perspective distortion.
DIST	Specifies distance from the viewer to the model origin. When used in conjunction with FIELD, distance * angle is directly proportional to picture size.
PART	Withhold some parts from the display.
COLO	Set/print colours for the background and the parts. Specifying a part range with null input prints the current colour setting.
ROTA	Rotate the model about the translated origin.
REST	Restore the model to its unrotated, untranslated state.
SHIF	Shift the model in the viewing plane.
SAVE	Save the current state of program parameters to a file.
RESE	Reset the current state of program parameters from a file.

^C the Terminate signal is used by *display* to abort the current command. Several repetitions will cause the program to EXIT.

>file The input is logged onto *file*. This file can be edited and used as a macro in subsequent runs.

<file The input is taken from *file* as if it were typed at the terminal.

When invoked with the operating system command '*rw display ...*', the command line arguments are passed to *display* as commands, one per line. In addition, shell escape access to the system with *cu*-like syntax is provided.

~!cmd Cmd is executed; its output goes to the terminal.

~$cmd Cmd is executed; its output goes to the program's input.

~>file Program output is written into *file* as well as the terminal.

~<file Program input is taken from *file*.

~. Quit.

A shell script *stereo* facilitates generation of stereo pairs. To initialize, '*stereo -I ang dist shif*' where 'ang' is the field angle, 'dist' is the distance to the model, and 'shif' is the shift needed to centre the model in one half of the screen. To use, '*stereo dist*' will return the commands needed to redraw a stereo pair at a distance 'dist'. The appropriate amount of perspective can thus be iteratively determined.

H. Device Drivers for Movie.BYU

These routines interface the device-independent parts of Movie.byu to the ITC graphics boards. The operations required include loading the colour look-up-tables (LUTs), clearing the screen, drawing a line, drawing a shaded raster line, and drawing alphanumeric text at a given location.

All of the ITC interface functions were developed in the C programming language. Functions written in C could be called from FORTRAN only if the subroutines which called them were compiled with '*f77 -e ...*', and if these subroutines were called without arguments.

Instead of adding an interface subroutine for each case, the device dependence was 'pushed up a level'. For example, the subroutine PLTLIN which would ordinarily call the subroutine PLTITC with its own arguments has been modified to update a common block with its arguments, then call the now argumentless PLTITC. PLTITC calls the function _line2() with PLTLIN's arguments which it extracts from the above common block.

_line2()

This function is a slightly modified version of the graphics library function *line2()*. Its arguments are in reverse order, and are pointers to integers rather than integers. This is to accommodate FORTRAN stacking and call-by-reference conventions.

_scanline(a)

This function writes the values contained in the integer array argument **a**[40] into the ITC frame buffer. The x and y coordinates and the length of the raster line are contained in the first three elements of this array.

clear(c)

This graphics library function sets the visible screen to the argument **c**.

type(string, x, y)

This function adjusts its x and y coordinate arguments to conform to the current field bounds, then calls the graphics library function *typestring()* to write the alphanumeric character array **string** to the frame buffer.

setlut(a)

This function loads the look-up-tables with the 256*3 integer values in its array argument **a**, and writes these values to a file named *maps/lutout*.

super_init(), super_line(), super_draw()

Movie.byu uses an RGB colour-cube model which, for 256 displayable colours, has a maximum dimension of 6x6x6. To compensate for the limited number of colour shades available, namely 6 in each of red, green, and blue (RGB),

---

[40] Arguments are *int* unless indicated otherwise.

an enhancement mode which selects 256 colours from the 2^{24} possible has been implemented [30]. This enhancement mode is based on a property of a three dimensional Peano curve.

A Peano curve has the property of traversing a space by making equal use of all the available degrees of freedom. A raster scan of an area, to take a 2D example, takes many more steps in the horizontal direction than in the vertical. As a result, two vertically adjacent pixels will always be sequentially far apart. A scan of an area following a Peano curve will take the same number of horizontal as vertical steps; the same number upward as downward, and leftward as rightward. This results in the curve staying relatively close to itself as it progresses. Very few adjacent points will be encountered sequentially far apart along the curve. This property ensures that points close together in the colour space will usually be close together in the resultant histogram.

Plate 1a shows the lowest-degree Peano traversal of a unit area - by dividing the area into quadrants. If each quadrant is further subdivided into quadrants, each of which is traversed in a similar manner, the curve shown in Plate 1c results. This curve is replicated four times at each level of the subdividing process.

After being subdivided two more times, the curve in Plate 1d would traverse each pixel in its unit area.

Plate 1b shows the lowest-degree Peano traversal of a unit volume - by dividing the volume into octants. By

National Library
of Canada

Canadian Theses Service

Bibliothèque nationale
du Canada

Service des thèses canadiennes

NOTICE

AVIS

THE QUALITY OF THIS MICROFICHE IS HEAVILY DEPENDENT UPON THE QUALITY OF THE THESIS SUBMITTED FOR MICROFILMING.

UNFORTUNATELY THE COLOURED ILLUSTRATIONS OF THIS THESIS CAN ONLY YIELD DIFFERENT TONES OF GREY.

LA QUALITE DE CETTE MICROFICHE DEPEND GRANDEMENT DE LA QUALITE DE LA THESE SOUMISE AU MICROFILMAGE.

MALHEUREUSEMENT, LES DIFFERENTES ILLUSTRATIONS EN COULEURS DE CETTE THESE NE PEUVENT DONNER QUE DES TEINTES DE GRIS.

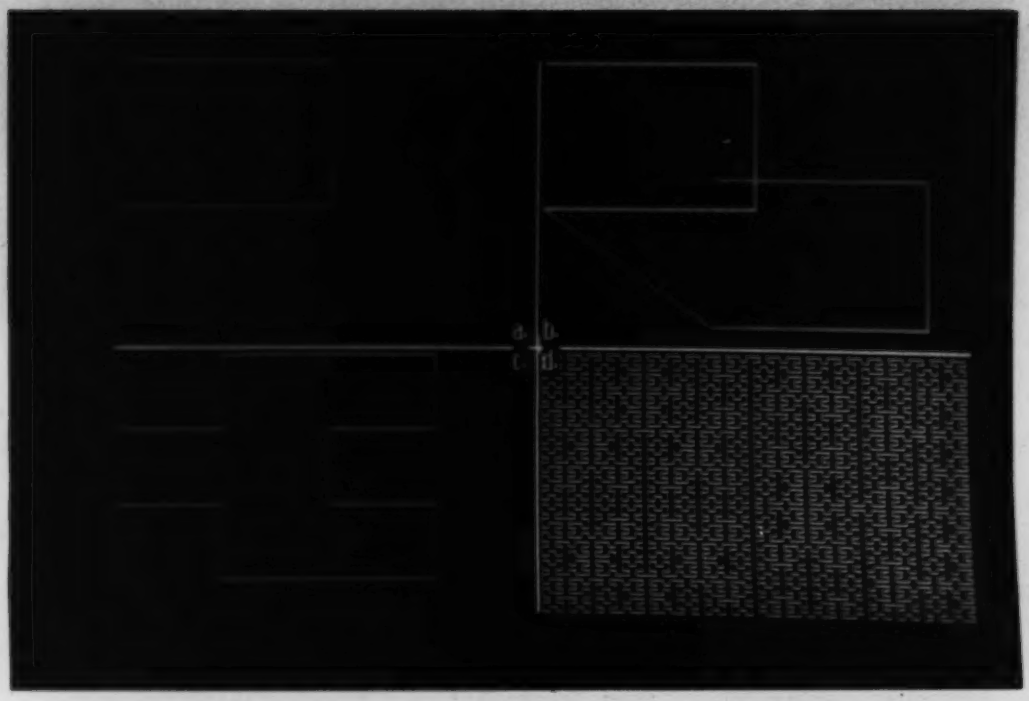

Plate 1. Peano Curves in 2 and 3 Dimensions

Plate 2. Peano Curve Traversal of a Colour Cube

analogy with the 2D case, a unit volume can be traversed voxel-by-voxel.

A Peano traversal of Movie.BYU's 6x6x6 colour space (red increases toward the right, blue increases downwards, green increases into the screen) is shown in Figure 2.

When this enhanced-colour mode is invoked, *super_init()* clears and opens the file */mnt/super.i*. The shaded raster lines are intercepted and written to this file a line at a time by the function *super_line()*. At the end of a frame, the file is read, the 8-bit RGB values are truncated to 6 bits and then histogrammed in a three dimensional array of 2^{18} elements. A three dimensional Peano curve traversal of this array yields a linear histogram [30] from which the 256 colours which best represent the original colour space are selected and loaded into the LUTs.

A second Peano curve traversal of the three dimensional array replaces each value with the LUT-address of the colour assigned to that point in the colour-space. The file is re-read and the truncated RGB values are used to index the modified three-dimensional array to determine the values of the raster output.

VII. Sample System Session

The following pages illustrate the use of the system. The operating syntax (enclosed in quotes) needed to execute the processes described in Chapters IV and VI are given. Photographs of the monitor show the resultant images.

Patient #1

In preparation for a new patient, a file system directory which will contain all of the patient's image and data files is created with *'mkdir dir'*. It becomes the current working directory through the command *'chdir dir'*. Within this directory, *'mkdir maps'* makes a directory for the LUT (look up table) setup files.

Enhance

The program *'enhance'* is invoked and used until further notice.

A video camera mounted over a light table is used to digitize the CT film images. Select the 'o' field using the command 'fo'. This ensures the images are acquired with maximum resolution. Set the boards for continuous acquisition with 'A' to position the CT image in the field of view of the camera[41]. Once positioned, 'G' grabs the current frame which can then be saved to a packed image file with 'sp '. This sequence is repeated for each CT image to be digitized.

---

[41] See Chapter IV-A for considerations to observe.

A complete set of digitized CT images is shown in Plate 3. A collage of images can be produced by shrinking each original image with '=qo', then cutting and pasting them into position with '@'. The macro '<multi.9' was used to produce Plate 3.

The command '!colours' was invoked, and the LUT setup file *false.4* loaded to produce the image of Plate 4. Within *colours*, the shades were translated, using 't', to achieve the best overall visual differentiation of detail. Even on cursory examination, the difference in perceptability of detail between Plate 3 and Plate 4 is readily apparent.

The images of Plate 5 are a false-coloured rendition of the four most interesting slices. Each of the four images are at half the original horizontal resolution, restored into the fields 'q','r','s',&'t', and then displayed in field 'o'.

Plate 6 shows a plot of the image histogram in blue, and a plot of the associated CDF (cumulative distribution function) in orange. These plots were overlaid on the CT image using the commands 'h bin cdf' and can be cleared without affecting the grey scale image by using 'c10'.

The LUT setup file *greys.B* is loaded via '!colours fr greys.B q'. Our convention of 64 grey shades overlaid by 3 graphics/text colours is demonstrated here.

Plate 7 shows a contrast-enhanced CT image. The boxcursor would typically be positioned over the central,

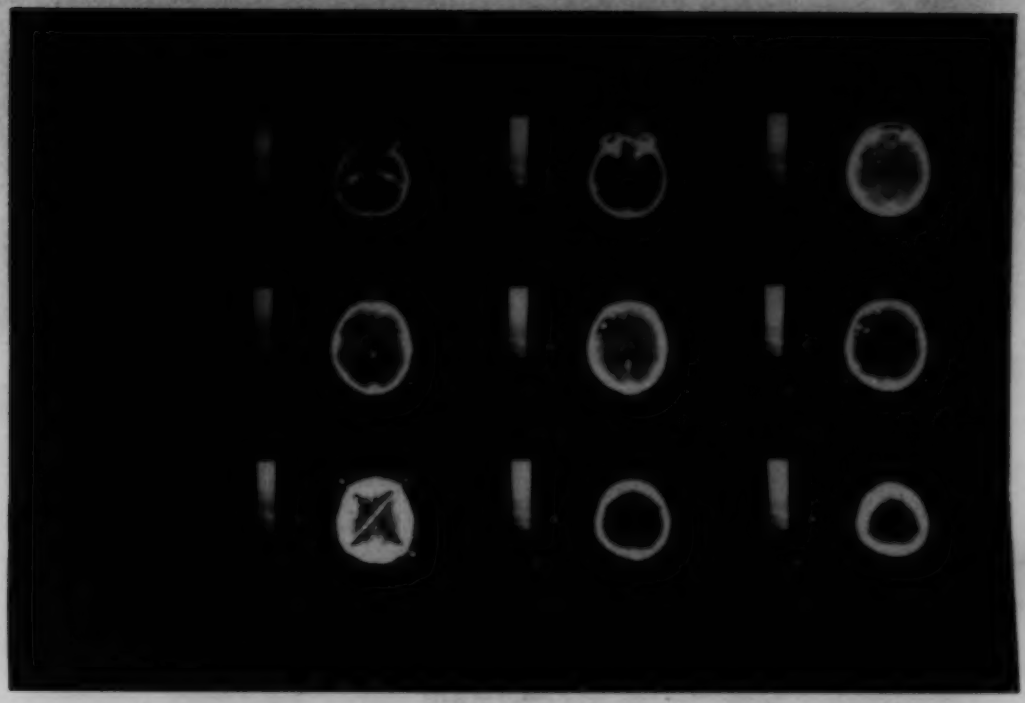

Plate 3. Complete Set of CT Images

Plate 4. Complete Set of CT Images - False Coloured

Plate 5. Four Selected CT Images - False Coloured

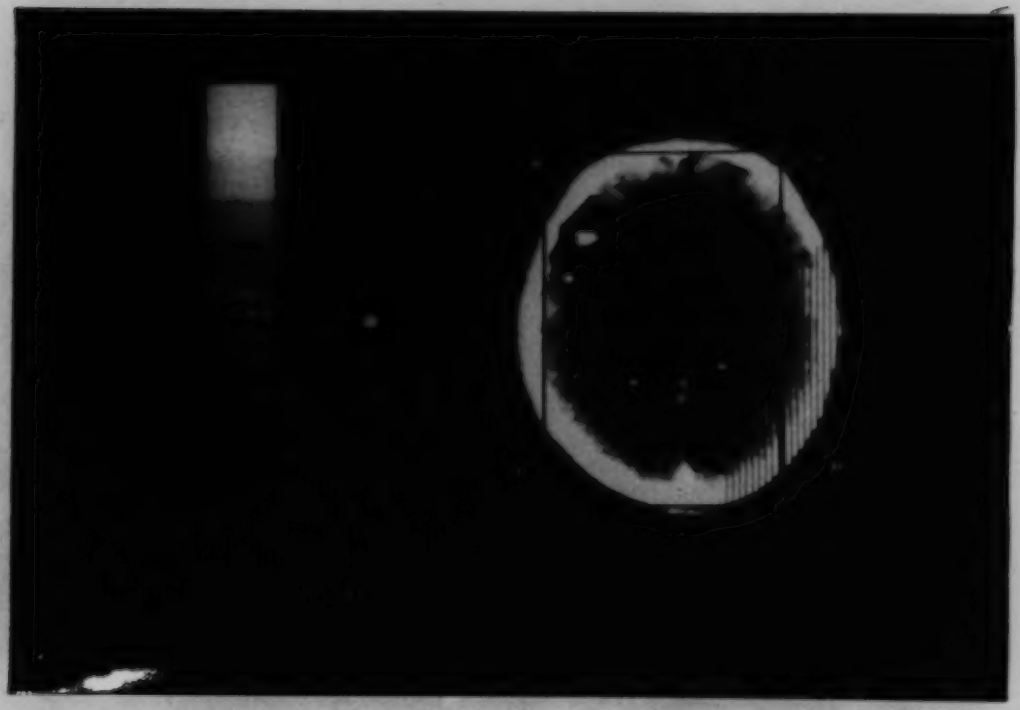

Plate 6. Single Greyscale CT Image - Histogrammed

relatively low contrast area of the image, followed by the commands 'h bin cdf'. The boxcursor is then repositioned as shown, and 'h flat bin cdf' executed. The command 'p ' puts the virtual image onto the screen. The increase in contrast over the image of Plate 6 is apparent over most of the image; only in some areas is there a loss of detail.

An edge-enhancing filter, invoked via 'F hi+i/3d' and 'p ', produced the image of Plate 8. The image has acquired a graininess, the result of increased noise due to the high-pass portion of the filter. Compared with Plate 6 however, many of the feature edges are sharper.

Plate 9 shows the result of "Difference-of-Gaussian" (DoG) filtering [17] using 'F dog/3.5'.

The left half of Plate 10 is the magnitude output of the Sobel edge detector used by the auto-contouring algorithm. Execute 'F edge 5 2 putemag' and 'p c10' to use this function. The right half of Plate 10 is the magnitude output of Kittler's absorption edge detector ('F edgea') which clearly exhibits less edge smearing which is precisely what he claimed for it.[24]

The output of the auto-contouring algorithm is shown in Plate 11 and Plate 12. Only the backgrounds vary. The contour tracing commands are in a macro file and can be executed by typing '<F3'. The relationship of the contours to the original image can be verified by Plate 11, while the

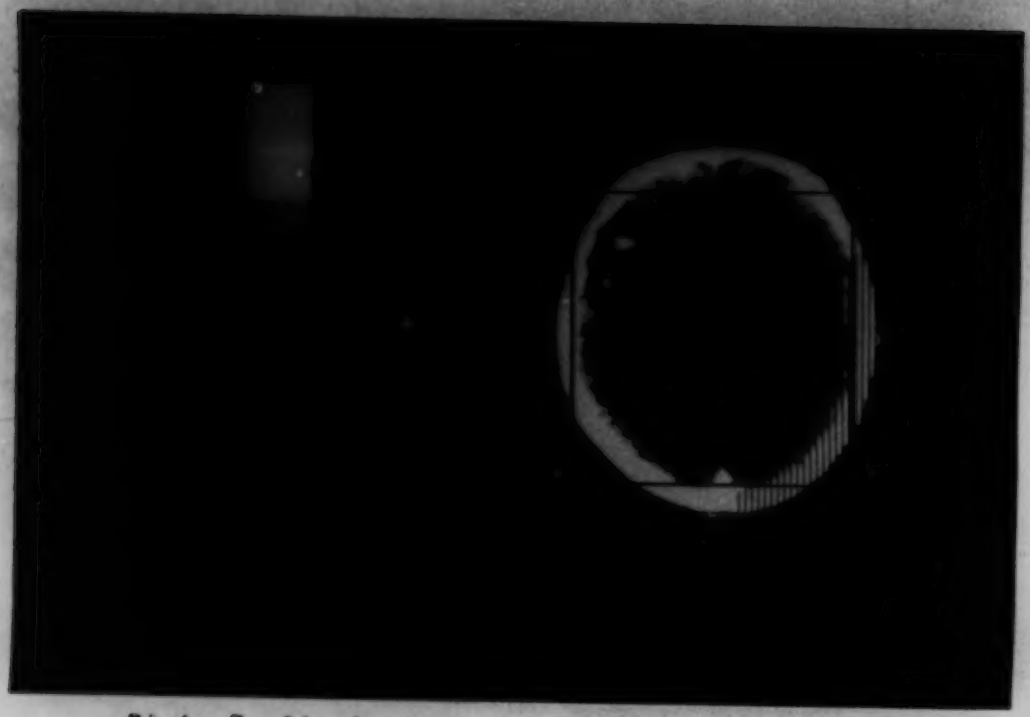

Plate 7. Single Contrast Enhanced CT Image

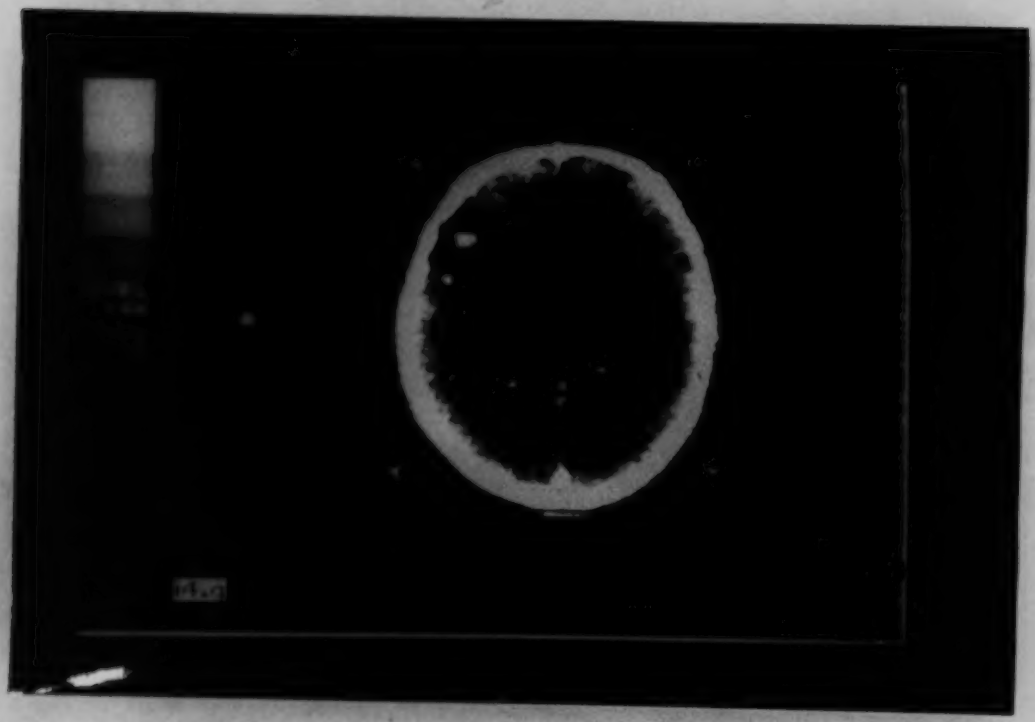

Plate 8. Edge-enhanced CT Image

Plate 9. Difference of Gaussian Filtered CT Image

Plate 10. Sobel/Absorption Edge Detected CT Image

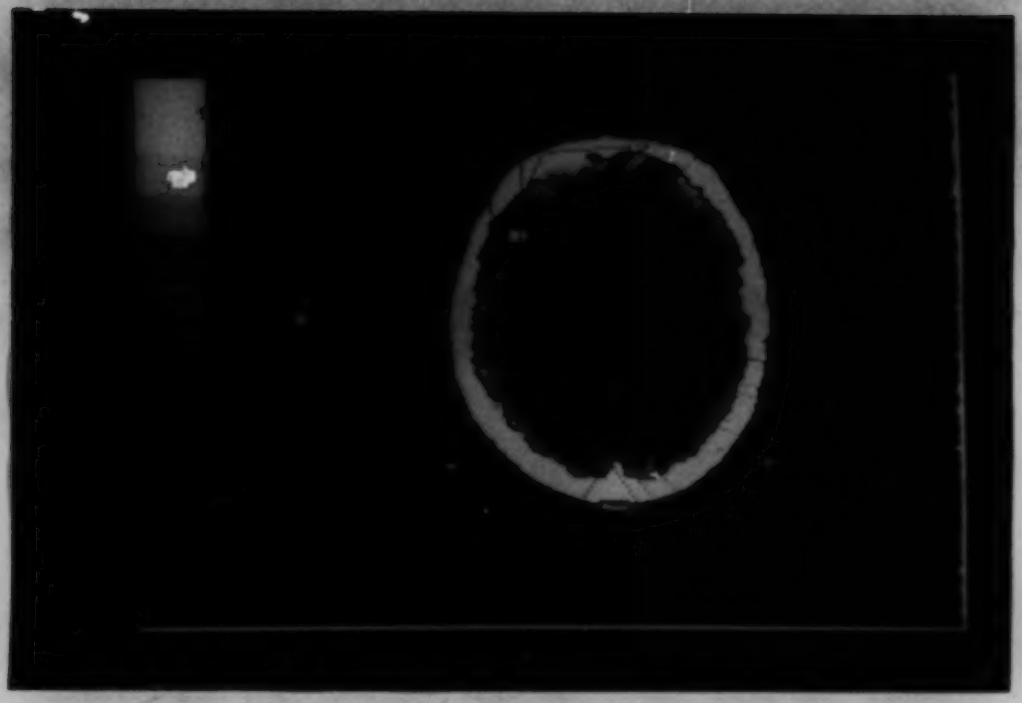

Plate 11. Auto-extracted Contours overlaid on CT image

Plate 12. Auto-extracted Contours on blank background

shapes of the contours are more easily seen in Plate 12. By 'F plot'-ing the contours onto two fields, one having a blank and the other a CT image background, both types of comparisons can easily be made.

To save the contour coordinates to a file, execute '&>file', 'F save', '&>'. After the contours have been extracted for each image and saved in files[42], *enhance* is exited.

Premos

The program *premos* is now invoked and used until further notice.

Contours can be manually entered, modified, or deleted as illustrated by Plate 13. Some areas of the image are un-traceable by the algorithm, and will require manual tracing by the operator. The usual procedure is to file-restore the auto-traced contours via 'fr file', edit them, add any required curves, and then 'fs file' on a slice-by-slice basis. The files will subsequently be loaded together and merged.

The locator pins of the stereotactic frame must be digitized in the manner and order shown in Plate 14. Subsequent registration operations, both inter-slice and between the model and the frame rely on this.

---

[42] A macro which performs this function is available.

A complete set of curves is shown in Plate 15 just prior to a file-save and the exiting of *premos*. (Note the small alignment errors at the vertices of the locator pins.)

Registrate and Mosaic

Executing '*registrate <file_pre >file_registered*' yields a file which can be examined by *premos*, as shown by Plate 16. Compare the alignment of the locator pin vertices with that of Plate 15. The change in aspect ratio is due to the difference between the width and the height of the monitor pixels. There is a similar difference in aspect ratio in the video camera, and so the two tend to cancel. When a geometrically regular polygon is displayed, it appears to be skewed.

Executing '*mosaic <file_registered*' will create the required *partn* and *geom* files without any operator interaction.

Display

The image of Plate 17 is the result of a number of *display* commands which appropriately position the model and the light sources relative to the viewer, rotate the model to a desired orientation, define the colours of each part, and update the screen. It can be quickly reproduced by reading the file *geom* doing a 'reset' from the file *demo.save*, and issuing a 'view' command. The yellow part is

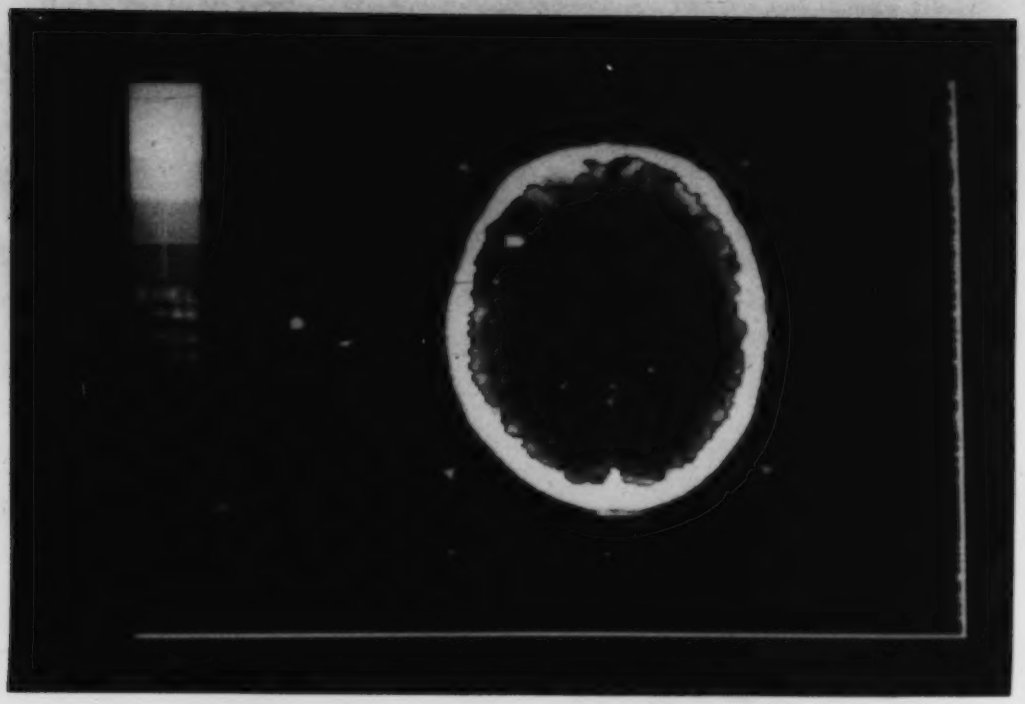

Plate 13. Manually Entered Contours

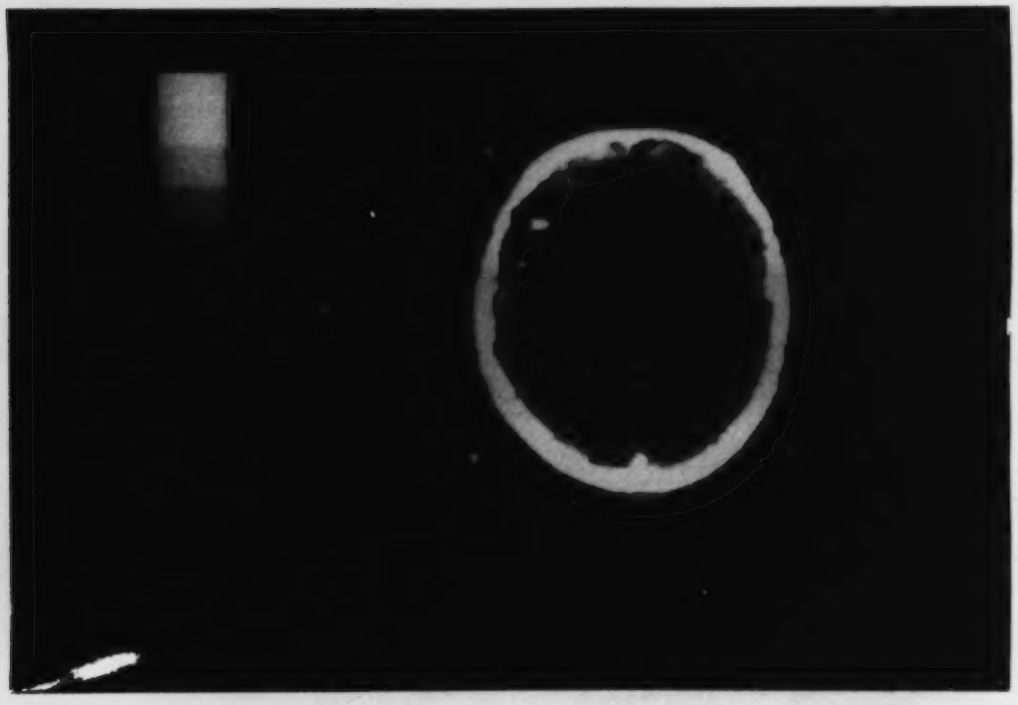

Plate 14. Premos Contours - Stereotactic Locator Pins

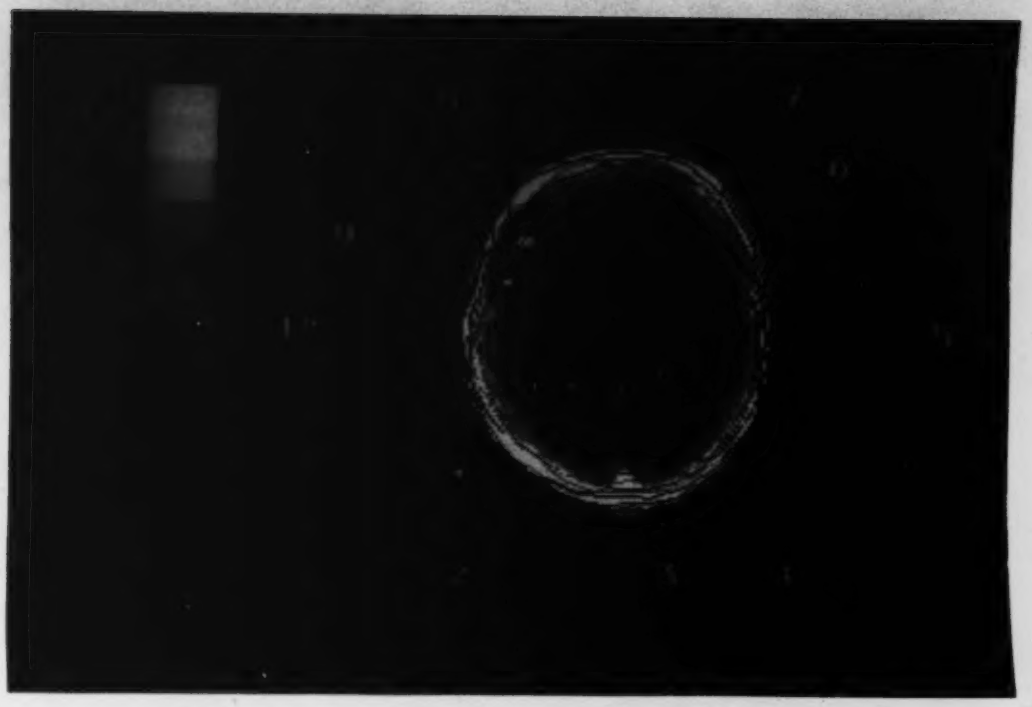

Plate 15. Premos Contours - Complete Set

Plate 16. Premos Contours - Auto Registered

Plate 17. Movie.BYU Display of 3D Model

Plate 18. Movie.BYU Display with Enhanced Colour

the hexagonal prism formed by the locator pins of the stereotactic frame. The blue part is the interior surface of the skull. The red part is part of the ventricular system.[43]

Plate 18 shows the enhanced colour version of Plate 15. Note the reduction of surface graininess and the recovery of subtle shading details. Note also that the yellow colour is affected considerably more than the blue or red by the increased resolution. This is due to the mixing of two colours (red and green) required to obtain yellow; the quantization errors are compounded.

While Plate 17 takes about one minute to be drawn in the low resolution mode shown, Plate 18 requires an additional two to three minutes. Macros can be used to produce such views non-interactively from interactively determined viewpoints and store them for future viewing.

Patient #2

A second set of CT images was recently obtained.

The image of Plate 19 is analagous to that of Plate 3. It was produced with the macro '<multi.9' In these images, the ventricles are much more clearly visible, and over a wider range of slices than in the previous set. In addition, a lesion is apparent in 7 of the 9 slices. On

---

[43] Display of the other parts was suppressed to reduce execution time during experimentation.

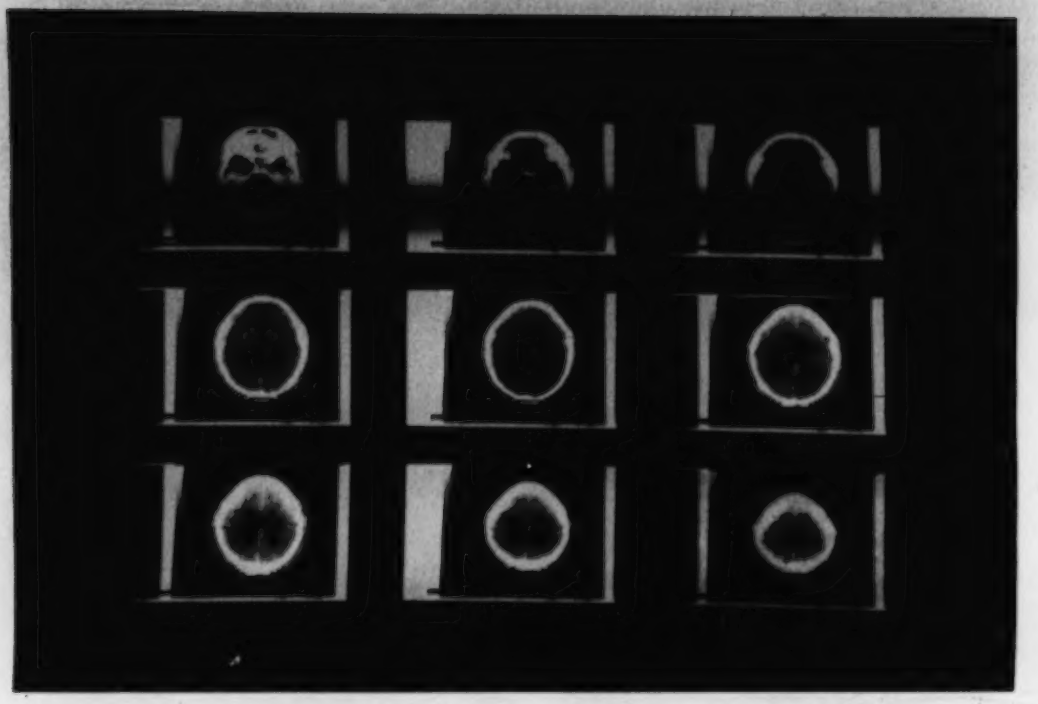

Plate 19. Complete set of CT images - Patient #2

Plate 20. Movie.BYU Display of 3D Model - 4 Views

slices 5, 6, and 7, it is clear that the lesion distends and nearly fills the ventricles on the right side.

Plate 20 shows four resultant *display* views at various model orientations.

- Upper Left (field 'q'): the model is seen from behind and below.
- Upper Right (field 'r'): the inverted model is seen from the front and above.
- Lower Left (field 's'): the model, lying on its right side, is seen from behind and above.
- Lower Right (field 't'): the model, oriented with its front toward the bottom of the screen is seen from directly above.

The views were written, one at a time, into the four low-resolution fields and then displayed simultaneously in the higher-resolution 'o' field. All parts except the tumour (red in colour) and the ventricles (green in colour) have been excluded from the display.

Examples of the uses of *'rw display'* are seen in Plates 21 and 22. A rudimentary on-screen menu is overlaid on a shaded surface view of the tumour and ventricles with an orange coloured wire-frame view of the inner surface of the skull behind them in Plate 21. The most commonly used *display* commands can thus be executed using the terminal's cursor keys and/or the mouse[44].

---

[44] These features were implemented at the very end of the project.

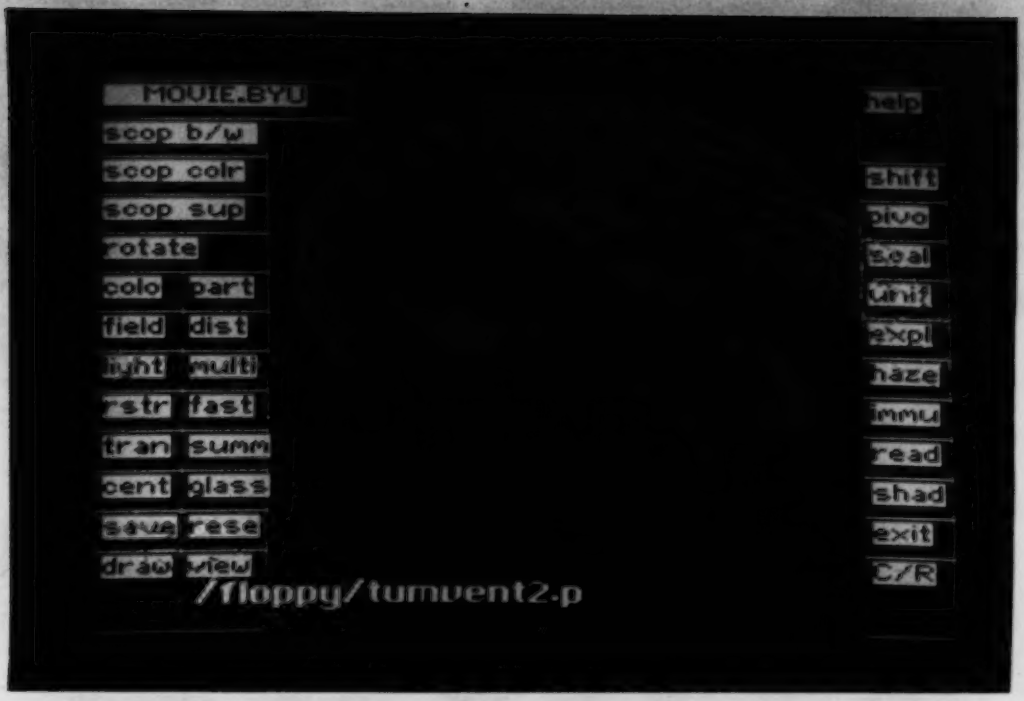

Plate 21. Movie.BYU Display of 3D Model - with menu

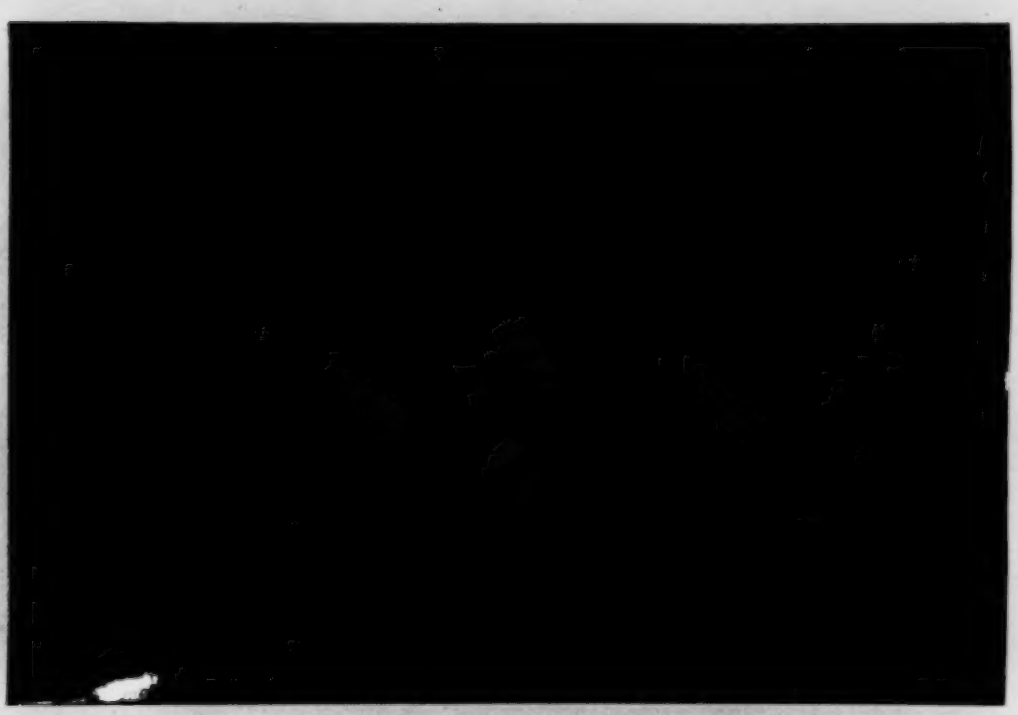

Plate 22. Movie.BYU Stereo Pair Display of 3D Model

The ability to invoke the operating system and direct its output into *display* makes the production of stereo image pairs, such as that of Plate 22, easier and less prone to error. The model must first be displayed so that it occupies only half of the screen. The initialization command '~!stereo -i 5 1000 35 view'[45] was followed by '~$stereo 900' to produce shaded surface stereo pair images.

---

[45] See Chapter VI-G, p.72 for details.

VIII. Conclusions

The aim of this project was to produce a semi-portable computer graphics system which could generate and display a three dimensional model of the essential information contained in a set of computed tomography (CT) scans of the brain as an aid to surgery.

This aim was only partly realized due to the complexity of the task which was not fully appreciated at the outset. The strategy adopted was to build a foundation of functions and programs which would address the details of manipulating images and graphic data. Higher level functions would then have been implemented, or commercial software installed to do more useful work. Virtually all the time spent went into attending to the details which seemed to never quite be adequately resolved.

Difficulties with the computer system itself were an obstacle to progress as well. Failure of one hard disk drive imposed stringent file space restrictions. Efforts to salvage part of the storage capacity through changes to the disk driver were met with success initially. Eventually, the drive had to be abandoned, and some of the seldom-used *Genix* system files were removed from the working disk to free up some space. Expert assistance in dealing with these problems was very limited.

Interface to the stereotactic frame was not achieved. With basic graphics functions just operational, attention to stereotactic details was not justified. From Rhodes [14] it

is seen that the problem is not a difficult one. The parameters of the frame might be obtained from the manufacturer or from the present stereotactic coordinate transformation computer programs in use in the Department of Surgery.

Dosimetry calculations and treatment simulations for intracranial radiation sources or for infrared thermal radiation delivered via optical fibre are not yet incorporated. Further work needs to be done in this area.

Ideally, the isodose surfaces could be calculated and superimposed on the 3D model of the pathology interactively. Optimization of source placement and orientation would attempt to achieve total tumour coverage with minimum exposure of healthy tissue. Current system performance would inhibit the interactive process. The images take relatively long to draw, and changes to the model geometry itself, as opposed to the viewpoint, are inherently non-interactive in Movie.byu.

At present, the system could not be used in a clinical setting. It may be a good base for further development and refinement, however.

References

[1] Flynn, M., Matteson, R., Dickie, D., Keyes, J. W., Jr., "Requirements for the display and analysis of three-dimensional medical image data", *SPIE Picture Archiving & Communication Systems (PACS II) for Medical Applications*, Vol.418,

[2] Herman, Gabor T. & Udupa Jayaram K., "Display of 3-D digital images: Computational foundations and medical applications", *IEEE Computer Graphics and Applications*, Vol.3, No.5, Aug.1983, pp.39-45

[3] Artzy, Ehud & Herman, Gabor T., "Boundary detection in 3-dimensions with a medical application", *Computer Graphics*, Vol.15, No.1, 1981, pp.92-123

[4] Batnitzky, Solomon M.D., Price, Hilton I. M.D., et al, "Three-dimensional computer reconstructions of brain lesions from surface contours provided by computed tomography: A prospectus", *Neurosurgery*, Vol.11, No.1, Part 1, 1982, pp.73-84

[5] Batnitzky, Solomon M.D., Price, Hilton I. M.D., et al, "Three-dimensional computer reconstruction in the study of brain lesions", *Automedica*, Vol.4, 1981, pp.37-50

[6] Lansky, Lester L., Batnizky, Solomon, Price, Hilton I., et al, "Application of three-dimensional computer reconstruction from computerized tomography to intracranial tumors in children", *Journal of Neuro-Oncology*, Vol.1, 1983, pp.347-356

[7] Perkins, W. J., & Green, R. J., "Three-dimensional reconstruction of biological sections", *Journal of Biomedical Engineering*, Vol.4, Jan.1982, pp.37-43

[8] McShan, D. L., Reinstein, L. E., Land, R. E., & Glicksman, A. S., "Automatic contour recognition in three-dimensional treatment planning", *Computed Tomography in Radiation Therapy*, 1983, pp.167-173

[9] McShan, D. L., Silverman, A., Lanza, D. M., Reinstein, L. E., & Glicksman, A. S., "A computerized three-dimensional treatment planning system utilizing interactive colour graphics", *British Journal of Radiology*, Vol.52, 1979, pp.478-481

[10] McShan, D. L., Haumann, D. R., Reinstein, L. E., & Glicksman, A. S., "An interactive three-dimensional radiation treatment planning system", *British Journal of Radiology (Suppl.)*, 15, 1979, pp.144-146

[11] Larsen, G. N., Glenn, W., Kishore, P. R. S., et al., "Computer processing of CT images: Advances and prospects", *Neurosurgery*, Vol.1, No.1, 1977, pp.73-79

[12] Cook, Larry T., Dwyer, S.J. III, et al, "A Three-dimensional display system for diagnostic imaging applications", *IEEE Computer Graphics and Applications*, Vol.3, No.5, Aug.1983, pp.13-19

[13] Christiansen, H., *MOVIE.BYU*, Brigham Young University, Provo, Utah 84602, 1983

[14] Rhodes, Michael L., Glenn, William V., Jr., Azzawi, Yu-Ming, "Computer graphics and an interactive stereotactic system for CT-aided neurosurgery", *IEEE Computer Graphics and Applications*, Vol.3, No.5,

[15] Kelly, Patrick J., Kall, Bruce A., Goerss, Stephan, & Alker, George J., "Stereotactic Surgery for CNS Tumours", *CT clinical symposium*, Vol.5, No.5, 1982

[16] Levialdi, S., "Finding the edge", *Digital Image Processing*, Simon, J. C. & Haralick, R. M. eds., D. Reidel, Dordrecht, 1981

[17] Marr, D. & Hildreth, E., "Theory of edge detection", *Proceedings of the Royal Society of London*, B207, 1980, pp.187-217

[18] Torre, Vincent & Poggio, Thomas, A., "On edge detection", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.PAMI-8, No.2, Mar.1986, pp.147-163

[19] Chen, Bor-Dong & Siy, Pepe, "Forward/backward contour tracing with feedback", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.PAMI-9, No.3, May 1987, pp.438-446

[20] Giraudon, G., "Edge detection from negative maximum of second derivative", *IEEE Conference on Computer Vision and Pattern Recognition*, Vol.85, 1985, pp.643-645

[21] Hashimoto, M. & Sklansky, J., "Edge detection by estimation of multiple-order derivatives", *IEEE Conference on Computer Vision and Pattern Recognition*, Vol.83, 1983, pp.318-325

[22] Healey, Glenn & Sanz, Jorge, L. C., "CONTAM: An Edge-based approach to segmenting images with irregular objects", *IEEE Conference on Computer Vision and Pattern Recognition*, Vol.85, 1985, pp.485-490

[23] Kittler, J., "On the accuracy of the Sobel edge detector", *Image and Vision Computing*, Vol.1, No.1, Feb.1983, pp.37-42

[24] Kittler, J. & Paler, K., "An absorption edge detector", *IEEE Conference on Computer Vision and Pattern Recognition*, Vol.83, 1983, pp.345-350

[25] Huertas, Andres & Medioni, Gerard, "Edge detection with subpixel precision", *IEEE Conference on Computer Vision and Pattern Recognition*, Vol.85, 1985, pp.633-636

[26] Nalwa, Vishvjit S., "Edge-detector resolution improvement by image interpolation", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.PAMI-9, No.3, May 1987, pp.446-451

[27] Verhagen, Frank, "Image processing for transillumination images", M.Sc. thesis, University of Alberta, Department of Electrical Engineering, 1985

[28] Imaging Technology Inc., "IP-512 Technical Manual - Multi-Bus Version 1.0" Nov.1982

[29] Foley, J. D., & Van Dam, A., *Fundamentals of Interactive Computer Graphics*, Addison-Wesley, Reading, MASS, 1982

[30] Lehar, A. F., & Stevens, R. J., "High-speed manipulation of the color chromaticity of digital amages", *IEEE Computer Graphics and Applications*, Vol.4, No.2, Feb.1984, pp.34-39

[31] Brown, Russell A., M.D., "A computerized tomography-computer graphics approach to stereotaxic localization", *Journal of Neurosurgery*, Vol.50, June 1979, pp.715-720

[32] Nagao, Makoto & Matsuyama, Takahashi, "Edge preserving smoothing", *IEEE International Joint Conference on Pattern Recognition*, Vol.78, 1978, pp.518-520

[33] Pavlidis, Theo & Wolberg, George, "An algorithm for the segmentation of bilevel images", *IEEE Conference on Computer Vision and Pattern Recognition*, Vol.86, 1986, pp.570-573

[34] Pavlidis, Theo, *Algorithms for Graphics and Image Processing*, Computer Science Press, Rockville, MD, 1982

[35] Rosenfeld, A. & Kak, A. C., *Digital Picture Processing*, 2nd ed., New York, Academic Press, 1982

[36] Wezska, Joan S., *Computer Graphics and Image Processing*, Vol.78, 1978, pp.259-265

[37] Yamaguchi, K., Kunii, T. L., & Fujimura, K., "Octree-related data structures and algorithms", *IEEE Computer Graphics and Applications*, Vol.4, No.1, Jan.1984, pp.53-59

Appendix 1

The following is a listing of the types of image processing filters that are available to the program *enhance*. Each of these entries is a directory in the *filterspecs* directory, and contains files whose contents define and initialize the convolution array or kernel.

/usr/graphics/filter/filterspecs/:

absorb/	gradient/	marr/	min/	replace
davies/	hi/	matsu	nagao	sobel/
dog/	hi+i/	max/	ripple/	transi,I,o,O
freichen/	lo/	median/	rmedian/	

The following is a listing of the filter filenames in each of the filter type directories.

./absorb:	135.1.4	135.2	135.E	45.1.4	45.2	45.E	
./davies:	3.x	3.y	5.x	5.y			
./dog:	1.3	1.5	3.5	3.7	5.7		
./freichen:	x	y					
./gradient:	5.x	5.y					
./hi:	3	3a	3b	3d			
./hi+i:	3	3a	3b	3d			
./lo:	10	13	19	20	3	31	5	9
./marr:	11						
./max:	3	5					
./median:	13	15	31	33	51	55	
./min:	min3	min5					

```
./prewitt:   3.x   3.y
./ripple:    135   45
./rmedian:   13    15   31   33   51   55
./sobel:     2.x   2.y  x    x.E  y    y.E
```

Appendix 2

The following is a listing of the contents of selected filtering files. The first lines contain the x and y convolution array dimensions and a scale factor by which the sum of the image pixel-kernel products is divided.

```
absorb/135.1.4
5 5 1000
 1414   1000   0      0   0
 1000   1414  1000   0   0
 0      1000   0  -1000   0
 0   0 -1000  -1414  1000
 0   0      0 -1000 -1414

absorb/135.2  (Kittler's)
5 5 3
 2   1   0   0   0
 1   2   1   0   0
 0   1   0  -1   0
 0   0  -1  -2  -1
 0   0   0  -1  -2

absorb/45.2   (Kittler's)
5 5 3
 0   0   0   1   2
 0   0   1   2   1
 0  -1   0   1   0
-1  -2  -1   0   0
-2  -1   0   0   0

absorb/45.E   (Kittler's)
#ifndef SCALE
#define SCALE 1
#endif  SCALE
#ifndef K
#define K 2
#endif  K
#define S SCALE
5 5 SCALE
  0    0    0    S    K
  0    0    S    K    S
  0   -S    0    S    0
 -S   -K   -S    0    0
 -K   -S    0    0    0

davies/3.x
3 3 1000
-464 0 464
-959 0 959
-464 0 464

davies/5.x
5 5 1000
00000 -0294 00000 00294 00000
-0582 -1000 00000 01000 00582
-1085 -1000 00000 01000 01085
-0582 -1000 00000 01000 00582
00000 -0294 00000 00294 00000

dog/1.3
3 3 9
-1 -1 -1
-1  8 -1
-1 -1 -1

dog/1.5
5 5 25
-1 -1 -1 -1 -1
-1 -1 -1 -1 -1
-1 -1 24 -1 -1
-1 -1 -1 -1 -1
-1 -1 -1 -1 -1

dog/3.5
5 5 300
-90 -90 -90 -90 -90
-90 160 160 160 -90
-90 160 160 160 -90
-90 160 160 160 -90
-90 -90 -90 -90 -90
5 5 225

dog/3.7
7 7 900
-90 -90 -90 -90 -90 -90 -90
-90 -90 -90 -90 -90 -90 -90
-90 -90 400 400 400 -90 -90
-90 -90 400 400 400 -90 -90
-90 -90 400 400 400 -90 -90
-90 -90 -90 -90 -90 -90 -90
-90 -90 -90 -90 -90 -90 -90
7 7 4410

gradient/5.x
5 5 12
-3 -4  0  4  3
-4 -6  0  6  4
-6 -12 0 12  6
-4 -6  0  6  4
-3 -4  0  4  3
```

```
hi+i/3
3 3 9
-1 -1 -1
-1 17 -1
-1 -1 -1

hi+i/3d
3 8
-1 -1 -1
-1 16 -1
-1 -1 -1

hi/3
3 3 9
-1 -1 -1
-1  8 -1
-1 -1 -1

lo/3
3 3 9
01 01 01
01 01 01
01 01 01

lo/31
3 1 3
01 01 01

marr/11
11 11 1200
   0   0   0  -1  -1  -2  -1
-1   0   0   0
   0   0  -2  -4  -8  -9  -8
-4  -2   0   0
   0  -2  -7 -15 -22 -23 -22
-15  -7  -2   0
  -1  -4 -15 -24 -14  -1 -14
-24 -15  -4  -1
  -1  -8 -22 -14  52 103  52
-14 -22  -8  -1
  -2  -9 -23  -1 103 178 103
-1 -23  -9  -1
  -1  -8 -22 -14  52 103  52
-14 -22  -8  -1
  -1  -4 -15 -24 -14  -1 -14
-24 -15  -4  -1
   0  -2  -7 -15 -22 -23 -22
-15  -7  -2   0
   0   0  -2  -4  -8  -9  -8
-4  -2   0   0
   0   0   0  -1  -1  -2  -1
-1   0   0   0

median/15
1 5 1
01
01
01
01
01

prewitt/3.x
3 3 6
-1  0  1
-1  0  1
-1  0  1

ripple/135
3 3 1000
00 -1000 1414
1000 00 -1000
-1414 1000 00

sobel/2.y
3 3 2
02 05 02
00 00 00
-2 -5 -2

sobel/x
3 3 1
-1 0 1
-2 0 2
-1 0 1

sobel/x.E
#ifndef SCALE
#define SCALE 1
#endif  SCALE
#ifndef K
#define K 2
#endif  K
#define S SCALE
3 3 SCALE
   -1  0  1
   -K  0  K
   -1  0  1
```

Appendix 3

The *Genix* command *mkfs /dev/fd0 mkfs.miniunix* creates a bootable (using 'bd', 'dc(4,0)/vmunix') floppy disk which holds a minimalist file system capable of supporting *Genix* without benefit of a hard disk drive. All files are standard LMC-supplied versions with the exception of:

- */reproduce/vmunix.mini*, (which was modified to make *bootdev=32 (/dev/fd0)*),
- the source files in */usr/sys/stand/them*, and
- the utilities in */stand* which supplant the original versions.

mkfs.miniunix:

```
usr/sys/mdec/flopboot.down
800 200
d--777 0 4
boot          ---644 0 4  /boot
vmunix        ---755 0 4  /reproduce/vmunix.mini
wboot.7.645       ---644 0 4  /usr/sys/mdec/wboot.7.645
wboot.8.512       ---644 0 4  /usr/sys/mdec/wboot.8.512
.profile      ---644 0 4  /.profile
bin       d--777 0 4
              chmod     ---755 0 4  /bin/chmod
              cp        ---755 0 4  /bin/cp
              ddt       ---755 0 4  /bin/ddt
              echo      ---755 0 4  /bin/echo
              ed        ---755 0 4  /bin/ed
              ls        ---755 0 4  /bin/ls
              mv        ---755 0 4  /bin/mv
              sh        ---755 0 4  /bin/sh
              stty      ---755 0 4  /bin/stty
              sync      ---755 0 4  /bin/sync
              tar       ---755 0 4  /bin/tar
              $
etc       d--777 0 4
              dumpdir       ---755 0 4  /etc/dumpdir
              init          ---755 0 4  /etc/init
```

```
            mount       ---755 0 4  /etc/mount
            restor      ---755 0 4  /etc/restor
            $
stand   d--777 0 4
            cpthem      ---755 0 4  /stand/cpthem
            formatthem  ---755 0 4  /stand/formatthem
            $
src  d--777 0 4
     Makefile     ---644 0 4 /usr/sys/stand/them/Makefile
     README       ---644 0 4 /usr/sys/stand/them/README
     cpthem.c     ---644 0 4 /usr/sys/stand/them/cpthem.c
     dcu.c        ---644 0 4 /usr/sys/stand/them/dcu.c
     formatthem.c ---644 0 4
/usr/sys/stand/them/formatthem.c
     libsa.a      ---644 0 4 /usr/sys/stand/them/libsa.a
     mkfs.stand   ---644 0 4 /usr/sys/stand/them/mkfs.stand
     $
usr     d--777 0 4
    include     d--777 0 4
        signal.h    ---644 0 4  /usr/include/signal.h
        sys         d--777 0 4
            dir.h       ---644 0 4  /usr/include/sys/dir.h
            filsys.h    ---644 0 4
/usr/include/sys/filsys.h
            ino.h       ---644 0 4  /usr/include/sys/ino.h
            inode.h     ---644 0 4  /usr/include/sys/inode.h
            param.h     ---644 0 4  /usr/include/sys/param.h
            saio.h      ---644 0 4  /usr/include/sys/saio.h
            $
        $
    $
mnt     d--777 0 4
            $
reproduce  d--777 0 4
           miniunix       ---755 0 4 /reproduce/miniunix
           mkfs.miniunix  ---644 0 4 /reproduce/mkfs.miniunix
           $
dev     d--777 0 4
            console         c--622  0   4   0   0
            mem             c--600  0   4   1   0
            kmem            c--600  0   4   1   1
            null            c--666  0   4   1   2
            kn0a            b--600  0   4   0   0
            kn0b            b--644  0   4   0   1
            kn0e            b--644  0   4   0   4
            fd0             b--066  0   4   0   32
            fd0v            b--066  0   4   0   34
            rfd0            c--600  0   4   2   32
            rkn0a           c--600  0   4   2   0
            rkn0b           c--600  0   4   2   1
            rkn0e           c--600  0   4   2   4
            tty             c--666  0   4   3   0
            tty07           c--622  0   4   0   0
```